```
LLL                 IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRR    TTTTTTTTTTTTTTTT   LLL
LLL                 IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRR    TTTTTTTTTTTTTTTT   LLL
LLL                 IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRR    TTTTTTTTTTTTTTTT   LLL
LLL                    III      BBB      BBB   RRR      RRR        TTT           LLL
LLL                    III      BBB      BBB   RRR      RRR        TTT           LLL
LLL                    III      BBB      BBB   RRR      RRR        TTT           LLL
LLL                    III      BBB      BBB   RRR      RRR        TTT           LLL
LLL                    III      BBB      BBB   RRR      RRR        TTT           LLL
LLL                    III      BBBBBBBBBBBB   RRRRRRRRRRR         TTT           LLL
LLL                    III      BBBBBBBBBBBB   RRRRRRRRRRR         TTT           LLL
LLL                    III      BBB      BBB   RRR  RRR            TTT           LLL
LLL                    III      BBB      BBB   RRR  RRR            TTT           LLL
LLL                    III      BBB      BBB   RRR    RRR          TTT           LLL
LLL                    III      BBB      BBB   RRR    RRR          TTT           LLL
LLL                    III      BBB      BBB   RRR    RRR          TTT           LLL
LLLLLLLLLLLLLLL     IIIIIIIII   BBBBBBBBBBBB   RRR      RRR        TTT           LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII   BBBBBBBBBBBB   RRR      RRR        TTT           LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII   BBBBBBBBBBBB   RRR      RRR        TTT           LLLLLLLLLLLLLLL
```

```
LL         IIIIII   BBBBBBBB    FFFFFFFFFF   IIIIII   NN      NN   CCCCCCCC   VV        VV   TTTTTTTTTT
LL         IIIIII   BBBBBBBB    FFFFFFFFFF   IIIIII   NN      NN   CCCCCCCC   VV        VV   TTTTTTTTTT
LL           II     BB      BB   FF            II     NN      NN   CC          VV      VV         TT
LL           II     BB      BB   FF            II     NN      NN   CC          VV      VV         TT
LL           II     BB      BB   FF            II     NNNN    NN   CC          VV      VV         TT
LL           II     BB      BB   FF            II     NNNN    NN   CC          VV      VV         TT
LL           II     BBBBBBBB    FFFFFFFF       II     NN  NN  NN   CC          VV      VV         TT
LL           II     BBBBBBBB    FFFFFFFF       II     NN  NN  NN   CC          VV      VV         TT
LL           II     BB      BB   FF            II     NN   NNNN    CC          VV      VV         TT
LL           II     BB      BB   FF            II     NN   NNNN    CC           VV    VV          TT
LL           II     BB      BB   FF            II     NN      NN   CC           VV    VV          TT
LL           II     BB      BB   FF            II     NN      NN   CC            VV  VV           TT
LLLLLLLLLL   IIIIII   BBBBBBBB   FF          IIIIII   NN      NN   CCCCCCCC      VV  VV           TT      ....
LLLLLLLLLL   IIIIII   BBBBBBBB   FF          IIIIII   NN      NN   CCCCCCCC       VVVV            TT      ....
                                                                                                        ....
                                                                                                        ....
LL         IIIIII   SSSSSSSS
LL         IIIIII   SSSSSSSS
LL           II     SS
LL           II     SS
LL           II     SS
LL           II     SS
LL           II       SSSSSS
LL           II       SSSSSS
LL           II           SS
LL           II           SS
LL           II           SS
LL           II           SS
LLLLLLLLLL   IIIIII   SSSSSSSS
LLLLLLLLLL   IIIIII   SSSSSSSS
```

```
    1   0001  0 %TITLE 'LIB$$FIND_CVT_PATH  for internal use of LIB$CVT_DX_DX'
    2   0002  0 MODULE LIB$$FIND_CVT_PATH (                    ! DFA of general data type conversion.
    3   0003  0                  IDENT = '1-006'              ! File:LIBFINCVT.B32 Edit: STAN1006
    4   0004  0                  ) =
    5   0005  1 BEGIN
    6   0006  1
    7   0007  1 !*******************************************************************************
    8   0008  1 !*                                                                             *
    9   0009  1 !*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
   10   0010  1 !*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
   11   0011  1 !*   ALL RIGHTS RESERVED.                                                      *
   12   0012  1 !*                                                                             *
   13   0013  1 !*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
   14   0014  1 !*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
   15   0015  1 !*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
   16   0016  1 !*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
   17   0017  1 !*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
   18   0018  1 !*   TRANSFERRED.                                                              *
   19   0019  1 !*                                                                             *
   20   0020  1 !*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
   21   0021  1 !*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
   22   0022  1 !*   CORPORATION.                                                              *
   23   0023  1 !*                                                                             *
   24   0024  1 !*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
   25   0025  1 !*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
   26   0026  1 !*                                                                             *
   27   0027  1 !*                                                                             *
   28   0028  1 !*******************************************************************************
   29   0029  1 !
   30   0030  1
   31   0031  1 !++
   32   0032  1 ! FACILITY:        General Utility Library
   33   0033  1 !
   34   0034  1 ! ABSTRACT:
   35   0035  1 !
   36   0036  1 !        This module contains LIB$$FIND_CVT_PATH routine which is called only
   37   0037  1 !        by LIB$CVT_DX_DX routine.  The reason that these two routines are in
   38   0038  1 !        different modules is because of anticipation of future updates to this
   39   0039  1 !        data conversion routines.  They are very large, and it is easier to
   40   0040  1 !        update them seperately.
   41   0041  1 !
   42   0042  1 ! ENVIRONMENT: User mode - AST reentrant
   43   0043  1 !
   44   0044  1 ! AUTHOR:          Farokh Morshed          01-09-1981
   45   0045  1 !
   46   0046  1 ! MODIFIED BY:
   47   0047  1 !
   48   0048  1 ! 1-001 - Original. FM1001      01-09-1981
   49   0049  1 ! 1-002 - Put in a check for DSC$W_LENGTH to be 1 when class A, or NCA, and
   50   0050  1 !         if class NCA stride must be 1.  FM 9-9-81
   51   0051  1 ! 1-003 - Put in a new data type, DSC$K_DTYPE_VT.  FM 1-DEC-81.
   52   0052  1 ! 1-004 - Put in a feature where DST_INFO [D_CEN] can be picked up for
   53   0053  1 !         LIB$CVT_DX_DX. FM 2-DEC-81.
   54   0054  1 ! 1-005 - Fix the bug that in [K_S_NLO, K_SD_NLO] negative inputs are picked
   55   0055  1 !         up as positive.  FM 1-Mar-83
   56   0056  1 ! 1-006 - Remove informational errors. STAN 24-Jul-1984.
   57   0057  1 !--
```

; 58          0058 1

```
  60        0059  1  %SBTTL 'Declarations'
  61        0060  1
  62        0061  1  ! SWITCHES:
  63        0062  1  !
  64        0063  1
  65        0064  1  SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
  66        0065  1
  67        0066  1  !+
  68        0067  1  ! LINKAGE
  69        0068  1  !-
  70        0069  1
  71        0070  1  LINKAGE
  72        0071  1      JSB_R1 = JSB (REGISTER = 0, REGISTER = 1) : PRESERVE (0, 1);
  73        0072  1
  74        0073  1  !
  75        0074  1  ! TABLE OF CONTENTS:
  76        0075  1  !
  77        0076  1
  78        0077  1  FORWARD ROUTINE
  79        0078  1      LIB$$FIND_CVT_PATH;                              ! Routine to find the conversion
  80        0079  1
  81        0080  1                                                      ! being done and report any
  82        0081  1                                                      ! unsupported fields in the descriptors.
  83        0082  1  !
  84        0083  1  ! INCLUDE FILES:
  85        0084  1  !
  86        0085  1
  87        0086  1  LIBRARY 'RTLSTARLE';                                ! System symbols, from SYS$LIBRARY:STARLET.L32
  88        0087  1
  89        0088  1  REQUIRE 'RTLIN:RTLPSECT';                           ! Define PSECT declarations macros
  90        0183  1
  91        0184  1  !
  92        0185  1  ! PSECTS:
  93        0186  1  !
  94        0187  1  DECLARE_PSECTS (LIB);                               ! Declare PSECTs for LIB$ facility
  95        0188  1  !
  96        0189  1  ! OWN STORAGE:
  97        0190  1  !
  98        0191  1  !     NONE
```

```
   100      0192  1  %SBTTL 'Deterministic Finite Automata for LIB$CVT_DX_DX'
   101      0193  1
   102      0194  1  GLOBAL ROUTINE LIB$$FIND_CVT_PATH (              ! Deterministic Finite Automata
   103      0195  1                                                  ! that will parse the source
   104      0196  1                                                  ! and destination descriptors.
   105      0197  1      SOURCE                                      ! Source descriptor that was passed
   106      0198  1                                                  ! to LIB$CVT_DX_DX.
   107      0199  1      , DESTINATION                               ! Destination descriptor that was
   108      0200  1                                                  ! passed to LIB$CVT_DX_DX.
   109      0201  1      , SRC_INFO                                  ! Address of a record that this
   110      0202  1                                                  ! routine will put the source
   111      0203  1                                                  ! information in.
   112      0204  1      , DST_INFO                                  ! Address of a record that this
   113      0205  1                                                  ! routine will put the destination
   114      0206  1                                                  ! information in.
   115      0207  1      , CVT_PATH                                  ! This code will determine what
   116      0208  1                                                  ! label of the LIB$CVT_DX_DX
   117      0209  1                                                  ! routine's CASE statement will
   118      0210  1                                                  ! be taken.
   119      0211  1      ) =
   120      0212  1
   121      0213  1  !++
   122      0214  1  !  FUNCTIONAL DESCRIPTION:
   123      0215  1  !
   124      0216  1  !        This routine is comprised of a Deterministic Finite Automaton, defined
   125      0217  1  !        as a 5 tuple :
   126      0218  1  !        STATES              : There is a state for each CLASS, and CLASS, DATA TYPE
   127      0219  1  !                              combination.
   128      0220  1  !        Alphabet            : Classes and Data types.
   129      0221  1  !        Mappings            : M(CLASS_S , DTYPE_B) := CLASS_S_DTYPE_B
   130      0222  1  !                                . . . . . . . . . . . . . . . . . .
   131      0223  1  !
   132      0224  1  !                              M(CLASS_D , DTYPE_W) := error . . . . .
   133      0225  1  !                                . . . . . . . . . . . . . . . . . .
   134      0226  1  !                                . . . . . . . . . . . . . . . . . .
   135      0227  1  !        Start state         :
   136      0228  1  !        Final states        : All possible combinations of CLASS, DTYPE.
   137      0229  1  !                              Some of these combinations are allowed, others
   138      0230  1  !                              are not. The error combinations are denoted by
   139      0231  1  !                              negative numbers as states.
   140      0232  1  !
   141      0233  1  !  MAINTENANCE OF THIS ROUTINE :
   142      0234  1  !
   143      0235  1  !This routine knows about all classes and data types of Appendix C V8.3.
   144      0236  1  !(You may want to update the above line everytime a change is made)
   145      0237  1  !To make an already existing CLASS, DATA TYPE combination a valid one, as
   146      0238  1  !opposed to an error you must :
   147      0239  1  !        1. Insert the symbol for that data type in DTYPE_TABLE in place of the
   148      0240  1  !           error state.
   149      0241  1  !        2. Define a FINAL_STATE for this combination.
   150      0242  1  !        3. Give it an action routine.
   151      0243  1  !
   152      0244  1  !To add a new data type you must :
   153      0245  1  !        1. Increment K_MAX_DATA_TYPES.
   154      0246  1  !        2. Set K_MAX_DTYPE_STA to value of the new data type.
   155      0247  1  !        3. Does any of the following need to be changed ?
   156      0248  1  !                a. K_SMLFINSTA
```

```
157    0249  1 |                          b. K_LRGFINSTA
158    0250  1 |                          c. K_TOP_SD
159    0251  1 |                          d. K_BOTTOM_SD
160    0252  1 |                      4. Define a new FINAL_STATE.
161    0253  1 |                      5. Each category in DTYPE_TABLE must have a new entry for the data type.
162    0254  1 |                         Note that the position (starting at 0) of each entry in each category is equivalent
163    0255  1 |                         to the data type value.
164    0256  1 |                      6. Add the new lable into the action routines CASE statement and
165    0257  1 |                         the sub-CASE statements in LIB$CVT_DX_DX will need to be modified to
166    0258  1 |                         include this new data type.
167    0259  1 |
168    0260  1 | To add a new class you must :
169    0261  1 |                      1. Increment K_MAX_CLASSES
170    0262  1 |                      2. Set K_MAX_CLASS_STA to value of the new class.
171    0263  1 |                      3. Increment K_ACTUAL_CLASSES.
172    0264  1 |                      4. Make a new K_STATEx_CLASS_y, where x is class value and y is the
173    0265  1 |                         symbol of the class.
174    0266  1 |                      5. Make a new FINAL_STATE.
175    0267  1 |                      6. Add a new category to the STATES structure at the end, with a index
176    0268  1 |                         value of one higher than the last category.
177    0269  1 |                      7. Make a new entry in CLASS_TABLE.
178    0270  1 |                      8. Make a new category in DTYPE_TABLE.
179    0271  1 |                      9. Make a new lable in the action routine CASE statement.
180    0272  1 |
181    0273  1 |
182    0274  1 | CALLING SEQUENCE:
183    0275  1 |
184    0276  1 |         ret_status.wlc.v = FIND_CVT_PATH ( SOURCE.rx.dx,
185    0277  1 |                                             DESTINATION.rx.dx,
186    0278  1 |                                             SRC_INFO.wr.r,
187    0279  1 |                                             DST_INFO.wr.r,
188    0280  1 |                                             CVT_PATH.wlu.r )
189    0281  1 |
190    0282  1 | FORMAL PARAMETERS:
191    0283  1 |
192    0284  1 |         SOURCE              Address of source descriptor passed to LIB$CVT_DX_DX.
193    0285  1 |         DESTINATION         Address of destination descriptor passed to LIB$CVT_DX_DX.
194    0286  1 |         SRC_INFO            Address of a record in LIB$CVT_DX_DX.
195    0287  1 |         DST_INFO            Address of a record in LIB$CVT_DX_DX.
196    0288  1 |         CVT_PATH            Address of a longword in LIB$CVT_DX_DX.
197    0289  1 |
198    0290  1 | IMPLICIT INPUTS:
199    0291  1 |
200    0292  1 |         NONE
201    0293  1 |
202    0294  1 | IMPLICIT OUTPUTS:
203    0295  1 |
204    0296  1 |         NONE
205    0297  1 |
206    0298  1 | COMPLETION STATUS: (or ROUTINE VALUE:)
207    0299  1 |
208    0300  1 |         K_UNSCLAROU                      : -1 Unsupported CLASS by routine.
209    0301  1 |         K_UNSDTYROU                      : -2 Unsupported DTYPE by routine.
210    0302  1 |         K_UNSDESROU                      : -3 Unsupported descriptor by routine.
211    0303  1 |         K_UNSDESSTA                      : -4 Unsupported descriptor by standard.
212    0304  1 |         K_UNSCLASTA                      : -5 Unsupported CLASS by standard.
213    0305  1 |         K_UNSDTYSTA                      : -6 Unsupported DTYPE by standard.
```

```
:  214      0306   1 |       K_INVNBDS                          : -7 Invalid NBDS because array size is greater
:  215      0307   1 |                                               than WU or dimension is not one.
:  216      0308   1 |       K_SUPPORTED                        :  1 This descriptor is supported.
:  217      0309   1 |
:  218      0310   1 | SIDE EFFECTS:
:  219      0311   1 |
:  220      0312   1 |     Caller of LIB$CVT_DX_DX must have LIB$EMULATE as a handler, if the
:  221      0313   1 |     source or destination descriptor explicitely ask for G, H, O conversions.
:  222      0314   1 |
:  223      0315   1 |--
:  224      0316   1
:  225      0317   2     BEGIN
:  226      0318   2
:  227      0319   2     BUILTIN
:  228      0320   2         CVTTP,
:  229      0321   2         CVTSP,
:  230      0322   2         CVTPT,
:  231      0323   2         CVTPS,
:  232      0324   2         CMPP;
:  233      0325   2
:  234      0326   2 !+
:  235      0327   2 ! MACRO
:  236      0328   2 !-
:  237      0329   2 !<BLF/MACRO>
:  238      0330   2
:  239      0331   2     MACRO
:  240      0332   2 !+
:  241      0333   2 ! These MACROs are used for clarity of code, since there is not builtin for them.
:  242      0334   2 !-
:  243    M 0335   2         CVTGH =
:  244      0336   2             LIB$$CVT_CVTGH_R1 %,
:  245      0337   2 !+
:  246      0338   2 ! These MACROs define portions of intermediate data buffer.
:  247      0339   2 !-
:  248    M 0340   2         LONG_1 =
:  249      0341   2             0, 0, 32, 0 %,
:  250    M 0342   2         LONG_2 =
:  251      0343   2             4, 0, 32, 0 %,
:  252    M 0344   2         LONG_3 =
:  253      0345   2             8, 0, 32, 0 %,
:  254    M 0346   2         LONG_4 =
:  255      0347   2             12, 0, 32, 0 %,
:  256    M 0348   2         LONG_5 =
:  257      0349   2             16, 0, 32, 0 %,
:  258    M 0350   2         LONG_6 =
:  259      0351   2             20, 0, 32, 0 %,
:  260    M 0352   2         LONG_7 =
:  261      0353   2             24, 0, 32, 0 %,
:  262    M 0354   2         LONG_8 =
:  263      0355   2             28, 0, 32, 0 %,
:  264    M 0356   2         S_LONG_1 =
:  265      0357   2             0, 0, 32, 1 %,
:  266    M 0358   2         S_LONG_2 =
:  267      0359   2             4, 0, 32, 1 %,
:  268    M 0360   2         S_BYTE_1 =
:  269      0361   2             0, 0, 8, 1 %,
:  270    M 0362   2         BYTE_1 =
```

```
 271        0363  2                     0, 0, 8, 0 %,
 272      M 0364  2             BYTE_2 =
 273        0365  2                     1, 0, 8, 0 %,
 274      M 0366  2             S_WORD_1 =
 275        0367  2                     0, 0, 16, 1 %,
 276      M 0368  2             WORD_1 =
 277        0369  2                     0, 0, 16, 0 %,
 278      M 0370  2             WORD_2 =
 279        0371  2                     2, 0, 16, 0 %,
 280      M 0372  2             NIBBLE_1 =
 281        0373  2                     0, 0, 4, 0 %,
 282        0374  2     !+
 283        0375  2     ! This MACRO calculates final states given the state and the token.
 284        0376  2     !-
 285      M 0377  2             FINAL_STATE (CLASS, DATA_TYPE) =
 286        0378  2                     CLASS*K_MAX_DATA_TYPES + DATA_TYPE %,
 287        0379  2     !+
 288        0380  2     ! This macro is used for SRC_INFO or DST_INFO scale field.
 289        0381  2     !-
 290      M 0382  2             M_SCALE =
 291        0383  2                     0, 0, 8, 1 %,
 292        0384  2     !+
 293        0385  2     ! This macro is used for SRC_INFO or DST_INFO length field.
 294        0386  2     !-
 295      M 0387  2             M_LEN =
 296        0388  2                     5, 0, 16, 0 %,
 297        0389  2     !+
 298        0390  2     ! Define the start state data structure of the DFA.
 299        0391  2     !-
 300      M 0392  2             START_STATE =
 301        0393  2                     VECTOR [K_MAX_CLASSES, BYTE, SIGNED] %;
 302        0394  2
 303        0395  2     !+
 304        0396  2     ! EXTERNAL
 305        0397  2     !-
 306        0398  2
 307        0399  2         EXTERNAL ROUTINE
 308        0400  2             LIB$STOP : NOVALUE,
 309        0401  2             CVTGH : JSB_R1 NOVALUE;
 310        0402  2
 311        0403  2     !+
 312        0404  2     ! These are the translation tables used when translating from or to packed decimal.
 313        0405  2     !-
 314        0406  2
 315        0407  2         EXTERNAL
 316        0408  2             LIB$AB_CVTTP_U,
 317        0409  2             LIB$AB_CVT_O_U,
 318        0410  2             LIB$AB_CVTTP_O,
 319        0411  2             LIB$AB_CVT_U_O,
 320        0412  2             LIB$AB_CVTPT_U,
 321        0413  2             LIB$AB_CVTPT_O,
 322        0414  2             LIB$AB_CVTPT_Z,
 323        0415  2             LIB$AB_CVTTP_Z;
 324        0416  2
 325        0417  2         EXTERNAL LITERAL                            ! Condition value symbols
 326        0418  2             LIB$_FATERRLIB;                         ! Fatal error in library.
 327        0419  2
```

```
328    0420  2
329    0421        !+
330    0422        ! FIELD DECLARATIONS
331    0423        !-
332    0424
333    0425           FIELD
334    0426              SRC_INFO_FIELDS =
335    0427                 SET
336    0428                 S_SCALE = [0, 0, 8, 1],
337    0429                 S_POINTER = [1, 0, 32, 0],
338    0430                 S_LEN = [5, 0, 16, 0],
339    0431                 S_SIGN = [7, 0, 1, 0]
340    0432                 TES;
341    0433
342    0434           FIELD
343    0435              DST_INFO_FIELDS =
344    0436                 SET
345    0437                 D_SCALE = [0, 0, 8, 1],
346    0438                 D_LEN = [5, 0, 16, 0]
347    0439                 TES;
348    0440
349    0441  2     !+
350    0442  2     ! Define some literals.
351    0443  2     !-
352    0444  2
353    0445  2        LITERAL
354    0446  2     !+
355    0447  2     !Status returned by FIND_CVT_PATH.
356    0448  2     !-
357    0449           K_UNSCLAROU = -1,            ! Unsupported CLASS by routine.
358    0450           K_UNSDTYROU = -2,            ! Unsupported DATA TYPE by routine.
359    0451           K_UNSDESROU = -3,            ! Unsupported descriptor by routine.
360    0452           K_UNSDESSTA = -4,            ! Unsupported descriptor by standard.
361    0453           K_UNSCLASTA = -5,            ! Unsupported CLASS by standard.
362    0454           K_UNSDTYSTA = -6,            ! Unsupported DTYPE by standard
363    0455           K_INVNBDS = -7,             ! Invalid NBDS
364    0456                                       ! because either array size is larger
365    0457                                       ! than a WU or it is not a one
366    0458                                       ! dimensional array.
367    0459           K_SUPPORTED = 1,            ! This descriptor is supported, and valid.
368    0460  2     !+
369    0461  2     !Some general values :
370    0462  2     !-
371    0463           K_INTMED_DATA_LENGTH = 32,      ! Intermediate data buffer length
372    0464           K_LRGST_QU = 65535,
373    0465           K_LRGST_LU = 4294967295,        ! Largest unsigned longword.
374    0466           K_LRGST_NEG_L = -2147483648,    ! Largest negative longword.
375    0467           K_LRGCLSSUP = DSC$K_CLASS_VS,   ! Largest CLASS supported by routine
376    0468           K_SMLCLSSUP = DSC$K_CLASS_S,    ! Smallest CLASS supported by routine
377    0469           K_MAX_DATA_TYPES = 38,          !Total number of DATA TYPES in the standard
378    0470           K_MAX_CLASSES = 15,             !Total number of classes supported,
379    0471                                           !Including the error case 0.
380    0472           K_MIN_CLASS = DSC$K_CLASS_S,    !Smalles class supported.
381    0473           K_MAX_CLASS = DSC$K_CLASS_VS,   !Largest class supported.
382    0474           K_MAX_CLASS_STA = DSC$K_CLASS_UBA,  !Max. class number supported by standard.
383    0475           K_MAX_DTYPE_STA = DSC$K_DTYPE_VT,   !Max. data type number supported by standard.
384    0476           K_ACTUAL_CLASSES = 6,           !Total classes that are allowed by the STATES table.
```

```
  385        0477   2            K_MSTNEGERR = -7,                                   !Most negative error state
  386        0478   2            K_SMLFINSTA = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_BU),          !Smallest final state supported.
  387        0479   2            K_LRGFINSTA = FINAL_STATE (DSCSK_CLASS_VS, DSCSK_DTYPE_VT),         !Largest final state supported.
  388        0480   2            K_TOP_SD = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_H), !Top state for class SD.
  389        0481   2            K_BOTTOM_SD = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_B),          !Bottom state for class SD.
  390        0482   2    !+
  391        0483   2    !These are the values of the members of K_ACTUAL_CLASSES :
  392        0484   2    !-
  393        0485   2            K_STATE1_CLASS_S = DSCSK_CLASS_S,
  394        0486   2            K_STATE2_CLASS_D = DSCSK_CLASS_D,
  395        0487   2            K_STATE4_CLASS_A = DSCSK_CLASS_A,
  396        0488   2            K_STATE9_CLASS_SD = DSCSK_CLASS_SD,
  397        0489   2            K_STATE10_CLASS_NCA = DSCSK_CLASS_NCA,
  398        0490   2            K_STATE11_CLASS_VS = DSCSK_CLASS_VS,
  399        0491   2    !+
  400        0492   2    !These are the final states that are valid CLASS, DATA TYPE combinations.
  401        0493   2    !The rest of the final states are error states.
  402        0494   2    !The first argument to the macro is CLASS, and the second is the DATA TYPE.
  403        0495   2    !-
  404        0496   2            K_S_BU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_BU),
  405        0497   2            K_S_WU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_WU),
  406        0498   2            K_S_LU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_LU),
  407        0499   2            K_S_B = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_B),
  408        0500   2            K_S_W = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_W),
  409        0501   2            K_S_L = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_L),
  410        0502   2            K_S_Q = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_Q),
  411        0503   2            K_S_F = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_F),
  412        0504   2            K_S_D = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_D),
  413        0505   2            K_S_T = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_T),
  414        0506   2            K_S_NU = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NU),
  415        0507   2            K_S_NL = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NL),
  416        0508   2            K_S_NLO = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NLO),
  417        0509   2            K_S_NR = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NR),
  418        0510   2            K_S_NRO = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NRO),
  419        0511   2            K_S_NZ = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_NZ),
  420        0512   2            K_S_P = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_P),
  421        0513   2            K_S_G = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_G),
  422        0514   2            K_S_H = FINAL_STATE (DSCSK_CLASS_S, DSCSK_DTYPE_H),
  423        0515   2            K_D_T = FINAL_STATE (DSCSK_CLASS_D, DSCSK_DTYPE_T),
  424        0516   2            K_A_BU = FINAL_STATE (DSCSK_CLASS_A, DSCSK_DTYPE_BU),
  425        0517   2            K_A_T = FINAL_STATE (DSCSK_CLASS_A, DSCSK_DTYPE_T),
  426        0518   2            K_SD_BU = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_BU),
  427        0519   2            K_SD_WU = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_WU),
  428        0520   2            K_SD_LU = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_LU),
  429        0521   2            K_SD_B = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_B),
  430        0522   2            K_SD_W = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_W),
  431        0523   2            K_SD_L = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_L),
  432        0524   2            K_SD_Q = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_Q),
  433        0525   2            K_SD_F = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_F),
  434        0526   2            K_SD_D = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_D),
  435        0527   2            K_SD_G = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_G),
  436        0528   2            K_SD_H = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_H),
  437        0529   2            K_SD_T = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_T),
  438        0530   2            K_SD_NU = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NU),
  439        0531   2            K_SD_NL = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NL),
  440        0532   2            K_SD_NLO = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NLO),
  441        0533   2            K_SD_NR = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NR),
```

```
442    0534  2              K_SD_NRO = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NRO),
443    0535  2              K_SD_NZ = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_NZ),
444    0536  2              K_SD_P = FINAL_STATE (DSCSK_CLASS_SD, DSCSK_DTYPE_P),
445    0537  2              K_NCA_BU = FINAL_STATE (DSCSK_CLASS_NCA, DSCSK_DTYPE_BU),
446    0538  2              K_NCA_T = FINAL_STATE (DSCSK_CLASS_NCA, DSCSK_DTYPE_T),
447    0539  2              K_VS_T = FINAL_STATE (DSCSK_CLASS_VS, DSCSK_DTYPE_T),
448    0540  2              K_VS_VT = FINAL_STATE (DSCSK_CLASS_VS, DSCSK_DTYPE_VT),
449    0541  2
450    0542  ! These are the left or right hand side of the conversion index.
451    0543  !-
452    0544               K_SMLINT = 1,
453    0545               K_LRGINT = 2,
454    0546               K_SMLFLT = 3,
455    0547               K_LRGFLT = 4,
456    0548               K_DEC = 5,
457    0549               K_NBDS = 6,
458    0550               K_TOT_CAT = 6;
459    0551
460    0552  !+
461    0553  ! Define two structures.
462    0554  ! START_STATE is just a vector of bytes, so we just use a macro to define it.
463    0555  ! STATES is a structure that we put all the states in other than the first state,
464    0556  ! and of course the final states and the states that never get used such as
465    0557  ! the states that contain non-supported CLASSes will not be in this structure.
466    0558  !-
467    0559
468    0560            STRUCTURE
469    0561              STATES [STATE, TOKEN] =
470    0562                [K_ACTUAL_CLASSES*K_MAX_DATA_TYPES]
471    0563  4            (STATES + (K_MAX_DATA_TYPES*
472    0564  5            BEGIN
473    0565  5
474    0566  5            CASE STATE FROM K_MIN_CLASS TO K_MAX_CLASS OF
475    0567  5              SET
476    0568  5
477    0569  5                [K_STATE1_CLASS_S] :
478    0570  5                    0;
479    0571  5
480    0572  5                [K_STATE2_CLASS_D] :
481    0573  5                    1;
482    0574  5
483    0575  5                [K_STATE4_CLASS_A] :
484    0576  5                    2;
485    0577  5
486    0578  5                [K_STATE9_CLASS_SD] :
487    0579  5                    3;
488    0580  5
489    0581  5                [K_STATE10_CLASS_NCA] :
490    0582  5                    4;
491    0583  5
492    0584  5                [K_STATE11_CLASS_VS] :
493    0585  5                    5;
494    0586  5
495    0587  5                [INRANGE, OUTRANGE] :
496    0588  6                    BEGIN
497    0589  6                    LIBSSTOP (LIBS_FATERRLIB);
498    0590  6                    0
```

```
  499   0591  5              END;
  500   0592  5          TES
  501   0593  5
  502   0594  5      END
  503   0595  5      ) + TOKEN)<0, %BPUNIT, 1>;
  504   0596
  505   0597  !+
  506   0598  2 ! This is the start state entries.
  507   0599  2 ! For each CLASS in the standard there is an entry here.  They are :
  508   0600  2 !     Z    ,S    ,D    ,V    ,A
  509   0601  2 !     ,P    ,none ,J    ,none ,SD
  510   0602  2 !     ,NCA  ,VS   ,VSA  ,UBS  ,UBA.
  511   0603  2 !-
  512   0604
  513   0605  2    BIND
  514   0606  2        CLASS_TABLE = UPLIT BYTE
  515   0607  2    %( Start state.  All classes. )%
  516   0608  2    (K_UNSCLAROU,DSCSK_CLASS_S,DSCSK_CLASS_D,K_UNSCLAROU,DSCSK_CLASS_A
  517   0609  2    ,K_UNSCLAROU,K_UNSCLASTA,K_UNSCLAROU,K_UNSCLASTA,DSCSK_CLASS_SD
  518   0610  2    ,DSCSK_CLASS_NCA,DSCSK_CLASS_VS,K_UNSCLAROU,K_UNSCLAROU,K_UNSCLAROU) : START_STATE;
  519   0611
  520   0612  2 !+
  521   0613  2 ! This is the rest of the state table.  It is seperate because of space efficiency
  522   0614  2 ! Each state contains entries for each data type supported by the standard.
  523   0615  2 ! Note that for space efficiency The final states are not in the vector.
  524   0616  2 ! Also since each state represents a supported CLASS, if a CLASS is not
  525   0617  2 ! supported ( by the standard or routine ), then the state has no entry in
  526   0618  2 ! the vector.  The index table for the vector will index to the proper place
  527   0619  2 ! in the vector below.
  528   0620  2 ! This table shows graphically what descriptors are valid.
  529   0621  2 !                                  DSCSK_DTYPE_x
  530   0622  2 !            BU WU LU B W L Q F D G H T NU NL NLO NR NRO NZ P VT
  531   0623  2 !DSCSK__CLASS__S    x  x  x  x x x x x x x x x x  x  x   x  x   x  x
  532   0624  2 !DSCSK__CLASS__D                           x
  533   0625  2 !DSCSK__CLASS__SD          x x x x x x x x x x  x  x   x  x   x
  534   0626  2 !DSCSK__CLASS__VS                          x                          x
  535   0627  2 !DSCSK__CLASS__A   x                       x
  536   0628  2 !DSCSK__CLASS__NCA x                       x
  537   0629  2 !
  538   0630  2 !
  539   0631  2 !Note that these data types are hard coded in ( zero based vector, and position
  540   0632  2 !of each data type is determined be the value of the symbol ) so if data type
  541   0633  2 !values are ever rearranged this table must be rearranged.
  542   0634  2 !-
  543   0635
  544   0636  2    BIND
  545   0637  2        DTYPE_TABLE = UPLIT BYTE
  546   0638  2    %( State zero.  Class z. )%
  547   0639  2    %( State one.   Class s. )%
  548   0640  2    (K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_BU,DSCSK_DTYPE_WU,DSCSK_DTYPE_LU
  549   0641  2    ,K_UNSDTYROU,DSCSK_DTYPE_B,DSCSK_DTYPE_W,DSCSK_DTYPE_L,DSCSK_DTYPE_Q
  550   0642  2    ,DSCSK_DTYPE_F,DSCSK_DTYPE_D,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_T
  551   0643  2    ,DSCSK_DTYPE_NU,DSCSK_DTYPE_NL,DSCSK_DTYPE_NLO,DSCSK_DTYPE_NR,DSCSK_DTYPE_NRO
  552   0644  2    ,DSCSK_DTYPE_NZ,DSCSK_DTYPE_P,K_UNSDTYROU,K_UNSDTYROU,K_UNSDESSTA
  553   0645  2    ,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_G,DSCSK_DTYPE_H,K_UNSDTYROU
  554   0646  2    ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU,K_UNSDTYROU
  555   0647  2    ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
```

```
556    0648   2        %( State two.  Class d. )%
557    0649   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
558    0650   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
559    0651   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,DSCSK_DTYPE_T
560    0652   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
561    0653   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
562    0654   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
563    0655   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
564    0656   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
565    0657   2        %( State three.  Class v. )%
566    0658   2        %( State four.  Class a. )%
567    0659   2        ,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_BU,K_UNSDESROU,K_UNSDESROU
568    0660   2        ,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU
569    0661   2        ,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_T
570    0662   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
571    0663   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDTYROU
572    0664   2        ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU
573    0665   2        ,K_UNSDESSTA,K_UNSDTYROU,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
574    0666   2        ,K_UNSDESSTA,K_UNSDESSTA
575    0667   2        %( State five.  Class p. )%
576    0668   2        %( State six.  Class 'undefined' )%
577    0669   2        %( State seven.  Class j. )%
578    0670   2        %( State eight.  Class 'undefined' )%
579    0671   2        %( State nine.  Class sd. )%
580    0672   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
581    0673   2        ,K_UNSDESSTA,DSCSK_DTYPE_B,DSCSK_DTYPE_W,DSCSK_DTYPE_L,DSCSK_DTYPE_Q
582    0674   2        ,DSCSK_DTYPE_F,DSCSK_DTYPE_D,K_UNSDESSTA,K_UNSDESSTA,DSCSK_DTYPE_T
583    0675   2        ,DSCSK_DTYPE_NU,DSCSK_DTYPE_NL,DSCSK_DTYPE_NLO,DSCSK_DTYPE_NR,DSCSK_DTYPE_NRO
584    0676   2        ,DSCSK_DTYPE_NZ,DSCSK_DTYPE_P,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
585    0677   2        ,K_UNSDESSTA,K_UNSDTYROU,DSCSK_DTYPE_G,DSCSK_DTYPE_H,K_UNSDESSTA
586    0678   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
587    0679   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
588    0680   2        %( State ten.  Class nca. )%
589    0681   2        ,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_BU,K_UNSDESROU,K_UNSDESROU
590    0682   2        ,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU,K_UNSDESROU
591    0683   2        ,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU,K_UNSDTYROU,DSCSK_DTYPE_T
592    0684   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
593    0685   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDTYROU
594    0686   2        ,K_UNSDTYROU,K_UNSDTYROU,K_UNSDESROU,K_UNSDESROU,K_UNSDTYROU
595    0687   2        ,K_UNSDESSTA,K_UNSDTYROU,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
596    0688   2        ,K_UNSDESSTA,K_UNSDESSTA
597    0689   2        %( State eleven.  Class vs. )%
598    0690   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
599    0691   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
600    0692   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,DSCSK_DTYPE_T
601    0693   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
602    0694   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
603    0695   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
604    0696   2        ,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA,K_UNSDESSTA
605    0697   2        ,K_UNSDESSTA,K_UNSDESSTA,DSCSK_DTYPE_VT
606    0698   2        %( State twelve.  Class vsa. )%
607    0699   2        %( State thirteen.  Class ubs. )%
608    0700   2        %( State fourteen.  Class uba. )%
609    0701   2        %( Add more states below )%
610    0702   2          ) : STATES;
611    0703   2
612    0704   2        LOCAL
```

```
613   0705   2            STATUS,                                          ! Status of this routine
614   0706   2            STATE,                                           ! State
615   0707   2            CLASS,                                           ! Current CLASS being looked at
616   0708   2            DTYPE,                                           ! Current DTYPE being looked at
617   0709   2            TOKEN,                                           ! The value of each data type supported
618   0710   2            LEFT_CVT : VOLATILE VECTOR [1],                  ! Left side of conversion index.
619   0711   2            RIGHT_CVT : VOLATILE VECTOR [1],                 ! Right side of conversion index.
620   0712   2            LEFT_OR_RIGHT_CVT : REF VECTOR,                  ! Left or right side of conversion index.
621   0713   2            SRC_OR_DST_INFO : REF BLOCK [, BYTE],            ! Source or destination info.
622   0714   2            SRC_OR_DST : REF BLOCK [, BYTE],                 ! Source or destination.
623   0715   2            TEMP_BUF : BLOCK [K_INTMED_DATA_LENGTH, BYTE];   ! Temporary buffer for reshuffling things.
624   0716   2
625   0717   2        MAP
626   0718   2            SOURCE : REF BLOCK [, BYTE],
627   0719   2            DESTINATION : REF BLOCK [, BYTE],
628   0720   2            SRC_INFO : REF BLOCK [, BYTE] FIELD (SRC_INFO_FIELDS),
629   0721   2            DST_INFO : REF BLOCK [, BYTE] FIELD (DST_INFO_FIELDS);
630   0722   2
631   0723   2     !+
632   0724   2     ! Traverse through the state table twice.  Once for source, and once for
633   0725   2     ! destination descriptor.
634   0726   2     ! Each time come up with a final state that indicates which left hand side
635   0727   2     ! (for the first traversing), or right hand side (for the second traversing) of
636   0728   2     ! conversion we have got, e.g. SMLINT, or LRGFLT, etc.
637   0729   2     ! The action codes also build SRC_INFO, and DST_INFO, and they do
638   0730   2     ! the conversions to the intermediate values.
639   0731   2     ! After we have the left hand side of conversion for source and the right hand
640   0732   2     ! side of conversion for destination
641   0733   2     ! descriptor, then stick them in a formula that maps these two into
642   0734   2     ! one final answer that indicates which general CLASS, DTYPE is being
643   0735   2     ! converted to which general CLASS, DTYPE, e.g. SMLINT_LRGFLT, or DEC_SMLFLT, etc.
644   0736   2     ! These final answers are the output parameter CVT_PATH that will end up as the
645   0737   2     ! index to the CASE statement in LIB$CVT_DX_DX.
646   0738   2     !-
647   0739   2     !+
648   0740   2     ! This loop is from 0 to 3, but we EXITLOOP at 2 because that is the second time
649   0741   2     ! through the loop and the end of the road.
650   0742   2     ! When the state table indicates an error, or we detect an error in an action routine,
651   0743   2     ! we will just EXITLOOP with the value given by the state table, or of our own choice.
652   0744   2     ! Note that we EXITLOOP when we detect errors in the action routines, e.g. if array
653   0745   2     ! size is greater than a WU.
654   0746   2     !-
655   0747   3        BEGIN
656   0748   4        STATUS = (INCRU TURN FROM 0 TO 3 DO
657   0749   5            BEGIN
658   0750   5     !+
659   0751   5     ! Determine CLASS and DTYPE of this go around, also set up LEFT_OR_RIGHT_CVT,
660   0752   5     ! and SRC_OR_DST, and SRC_OR_DST_INFO.
661   0753   5     ! If this is the third time through this loop, we are finished.
662   0754   5     !-
663   0755   5
664   0756   5            CASE .TURN FROM 0 TO 2 OF
665   0757   5                SET
666   0758   5
667   0759   5                [0] :
668   0760   6                    BEGIN
669   0761   6                    CLASS = .SOURCE [DSC$B_CLASS];
```

```
670    0762  6                          DTYPE = .SOURCE [DSC$B_DTYPE];
671    0763  6                          SRC_OR_DST = .SOURCE;
672    0764  6                          SRC_OR_DST_INFO = .SRC_INFO;
673    0765  6                          LEFT_OR_RIGHT_CVT = LEFT_CVT;
674    0766  6                          END;
675    0767  5
676    0768  5                      [1] :
677    0769  6                          BEGIN
678    0770  6                          CLASS = .DESTINATION [DSC$B_CLASS];
679    0771  6                          DTYPE = .DESTINATION [DSC$B_DTYPE];
680    0772  6                          SRC_OR_DST = .DESTINATION;
681    0773  6                          SRC_OR_DST_INFO = .DST_INFO;
682    0774  6                          LEFT_OR_RIGHT_CVT = RIGHT_CVT;
683    0775  6                          END;
684    0776  5
685    0777  5                      [2] :
686    0778  5                          EXITLOOP K_SUPPORTED;
687    0779  5                      TES;
688    0780  5
689    0781  5      !+
690    0782  5      ! Filter out the out-of-range CLASS and DTYPE.
691    0783  5      !-
692    0784  5
693    0785  5              IF .CLASS GTRU K_MAX_CLASS_STA THEN EXITLOOP K_UNSCLASTA;
694    0786  5
695    0787  5              IF .DTYPE GTRU K_MAX_DTYPE_STA THEN EXITLOOP K_UNSDTYSTA;
696    0788  5
697    0789  5      !+
698    0790  5      ! Crank up the finite state machine. start looking in the start state.
699    0791  5      !-
700    0792  5              STATE = .CLASS_TABLE [.CLASS];
701    0793  5      !+
702    0794  5      ! Action code for each state that results from the start state.
703    0795  5      !-
704    0796  5
705    0797  5              CASE .STATE FROM K_MSTNEGERR TO K_LRGCLSSUP OF
706    0798  5                  SET
707    0799  5
708    0800  5                  [K_INVNBDS TO K_UNSCLAROU] :
709    0801  5                      EXITLOOP .STATE;                        ! Exit the INCR with the error
710    0802  5                                                             ! resulted from the start state.
711    0803  5
712    0804  5                  [K_SMLCLSSUP TO K_LRGCLSSUP] :
713    0805  6                      BEGIN
714    0806  6                      TOKEN = .DTYPE_TABLE [.STATE, .DTYPE];  ! This is a final state, but
715    0807  6                                                             ! some constants need to be
716    0808  6                                                             ! Applied to it yet.
717    0809  6                                                             ! This is just a data type, or a negative number if error.
718    0810  6
719    0811  6                      IF .TOKEN LSS 0 THEN EXITLOOP .TOKEN;   ! Exit INCR with the error resulted
720    0812  6                                                             ! in a final state.
721    0813  6
722    0814  6                      STATE = FINAL_STATE (.STATE, .TOKEN);   ! Find the final state.
723    0815  6                      END;
724    0816  5
725    0817  5                  [INRANGE, OUTRANGE] :
726    0818  5                      LIBSSTOP (LIBS_FATERRLIB);
```

```
727   0819  5                          TES;
728   0820  5
729   0821  5         !
730   0822  5         ! This CASE statement contains the action code for each final state other than
731   0823  5         ! the erro⁻ states.
732   0824  5         ! The caller of this routine has set up the pointer and length of SRC_INFO
733   0825  5         ! to be the intermediate data area (INTMED_DATA), so in the CASE below we
734   0826  5         ! will change pointer and length if needed (e.g. any NBDS), otherwise we never
735   0827  5         ! touch it.
736   0828  5         ! If .TURN is 0 then we are processing the left side of the conversion, when
737   0829  5         ! it is 1 we are processing the right side of the conversion. Another words
738   0830  5         ! if .TURN is 0 we are looking at the CLASS, DATA TYPE of source, and if
739   0831  5         ! it is 1 we are looking at CLASS, DATA TYPE of destination.
740   0832  5         ! These action codes determine which category (e.g. K_SMLINT or K_DEC as
741   0833  5         ! described in LIBSCVT_DX_DX documentation) source or destination data type
742   0834  5         ! falls into.  They also convert the source data type to an intermediate
743   0835  5         ! data type.  For more detail refer to the functional description of
744   0836  5         ! LIBSCVT_DX_DX.
745   0837  5         !-
746   0838  5
747   0839  5                  CASE .STATE FROM K_SMLFINSTA TO K_LRGFINSTA OF
748   0840  5                      SET
749   0841  5
750   0842  5                      [K_S_BU, K_SD_BU] :
751   0843  6                          BEGIN
752   0844  6                          .LEFT_OR_RIGHT_CVT = K_SMLINT;
753   0845  6
754   0846  6                          IF .TURN EQL 0
755   0847  6                          THEN
756   0848  6                              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 8, 0;,
757   0849  6                                  BYTE];
758   0850  6
759   0851  5                          END;
760   0852  5
761   0853  5                      [K_S_WU, K_SD_WU] :
762   0854  6                          BEGIN
763   0855  6                          .LEFT_OR_RIGHT_CVT = K_SMLINT;
764   0856  6
765   0857  6                          IF .TURN EQL 0
766   0858  6                          THEN
767   0859  6                              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 16, 0;,
768   0860  6                                  BYTE];
769   0861  6
770   0862  5                          END;
771   0863  5
772   0864  5                      [K_S_LU, K_SD_LU] :
773   0865  6                          BEGIN
774   0866  6                          .LEFT_OR_RIGHT_CVT = K_LRGINT;
775   0867  6
776   0868  6                          IF .TURN EQL 0
777   0869  6                          THEN
778   0870  6                              .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;,
779   0871  6                                  BYTE];
780   0872  6
781   0873  5                          END;
782   0874  5
783   0875  5                      [K_S_B, K_SD_B] :
```

```
784   0876  6                              BEGIN
785   0877  6                              .LEFT_OR_RIGHT_CVT = K_SMLINT;
786   0878  6
787   0879  6                              IF .STATE EQL K_SD_B THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
788   0880  6
789   0881  6                              IF .TURN EQL 0
790   0882  6                              THEN
791   0883  6                                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSCSA_POINTER], 0, 0, 8, 1;,
792   0884  6                                      BYTE];
793   0885  6
794   0886  6                              END;
795   0887  6
796   0888  5                          [K_S_W, K_SD_W] :
797   0889  6                              BEGIN
798   0890  6                              .LEFT_OR_RIGHT_CVT = K_SMLINT;
799   0891  6
800   0892  6                              IF .STATE EQL K_SD_W THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
801   0893  6
802   0894  6                              IF .TURN EQL 0
803   0895  6                              THEN
804   0896  6                                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSCSA_POINTER], 0, 0, 16, 1;,
805   0897  6                                      BYTE];
806   0898  6
807   0899  5                              END;
808   0900  5
809   0901  5                          [K_S_L, K_SD_L] :
810   0902  6                              BEGIN
811   0903  6                              .LEFT_OR_RIGHT_CVT = K_SMLINT;
812   0904  6
813   0905  6                              IF .STATE EQL K_SD_L THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
814   0906  6
815   0907  6                              IF .TURN EQL 0
816   0908  6                              THEN
817   0909  6                                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSCSA_POINTER], 0, 0, 32, 1;,
818   0910  6                                      BYTE];
819   0911  6
820   0912  5                              END;
821   0913  5
822   0914  5                          [K_S_Q, K_SD_Q] :
823   0915  6                              BEGIN
824   0916  6                              .LEFT_OR_RIGHT_CVT = K_LRGINT;
825   0917  6
826   0918  6                              IF .STATE EQL K_SD_Q THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
827   0919  6
828   0920  6                              IF .TURN EQL 0
829   0921  6                              THEN
830   0922  7                                  BEGIN
831   0923  7                                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSCSA_POINTER], 0, 0, 32, 0;, BYTE];
832   0924  7                                  (.SRC_INFO [S_POINTER] + 4) = .BLOCK [.SOURCE [DSCSA_POINTER] + 4, 0, 0, 32, 0;, BYTE];
833   0925  7
834   0926  7                                  IF .BLOCK [.SRC_INFO [S_POINTER], 4, 31, 1, 0;, BYTE]
835   0927  7                                  THEN
836   0928  8                                      BEGIN
837   0929  8                                      .SRC_INFO [S_POINTER] = ..SRC_INFO [S_POINTER] XOR %X'FFFFFFFF';
838   0930  8                                      .SRC_INFO [S_POINTER] + 4 = .(.SRC_INFO [S_POINTER] + 4) XOR %X'FFFFFFFF';
839   0931  8
840   0932  8                                      IF ..SRC_INFO [S_POINTER] EQLU K_LRGST_LU
```

```
841    0933  8                          THEN
842    0934  9                              BEGIN
843    0935  9                              .SRC_INFO [S_POINTER] = 0;
844    0936  9                              .SRC_INFO [S_POINTER] + 4 = .(.SRC_INFO [S_POINTER] + 4) + 1;
845    0937  9                              END
846    0938  8                          ELSE
847    0939  8                              .SRC_INFO [S_POINTER] = ..SRC_INFO [S_POINTER] + 1;
848    0940  8
849    0941  8                          SRC_INFO [S_SIGN] = 1;
850    0942  7                          END;
851    0943  7
852    0944  6                      END;
853    0945  6
854    0946  5              END;
855    0947  5
856    0948  5          [K_S_F, K_SD_F] :
857    0949  6              BEGIN
858    0950  6              .LEFT_OR_RIGHT_CVT = K_SMLFLT;
859    0951  6
860    0952  6              IF .STATE EQL K_SD_F THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
861    0953  6
862    0954  6              IF .TURN EQL 0
863    0955  6              THEN
864    0956  6                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;.
865    0957  6                      BYTE];
866    0958  6
867    0959  5              END;
868    0960  5
869    0961  5          [K_S_D, K_SD_D] :
870    0962  6              BEGIN
871    0963  6              .LEFT_OR_RIGHT_CVT = K_SMLFLT;
872    0964  6
873    0965  6              IF .STATE EQL K_SD_D THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
874    0966  6
875    0967  6              IF .TURN EQL 0
876    0968  6              THEN
877    0969  7                  BEGIN
878    0970  7                  .SRC_INFO [S_POINTER] = .BLOCK [.SOURCE [DSC$A_POINTER], 0, 0, 32, 0;. BYTE];
879    0971  7                  (.SRC_INFO [S_POINTER] + 4) = .BLOCK [.SOURCE [DSC$A_POINTER] + 4, 0, 0, 32, 0;. BYTE];
880    0972  6                  END;
881    0973  6
882    0974  5              END;
883    0975  5
884    0976  5          [K_S_G, K_SD_G] :
885    0977  6              BEGIN
886    0978  6              .LEFT_OR_RIGHT_CVT = K_LRGFLT;
887    0979  6
888    0980  6              IF .STATE EQL K_SD_G THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
889    0981  6
890    0982  6              IF .TURN EQL 0 THEN CVTGH (.SOURCE [DSC$A_POINTER], .SRC_INFO [S_POINTER]);
891    0983  6
892    0984  5              END;
893    0985  5
894    0986  5          [K_S_H, K_SD_H] :
895    0987  6              BEGIN
896    0988  6              .LEFT_OR_RIGHT_CVT = K_LRGFLT;
897    0989  6
```

```
  898      0990  6                              IF .STATE EQL K_SD_H THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
  899      0991  6
  900      0992  6                              IF .TURN EQL 0 THEN CH$MOVE (16, .SOURCE [DSC$A_POINTER], .SRC_INFO [S_POINTER]);
  901      0993  6
  902      0994  5                              END;
  903      0995  5
  904      0996  5                          [K_S_T, K_SD_T] :
  905      0997  6                              BEGIN
  906      0998  6                              .LEFT_OR_RIGHT_CVT = K_NBDS;
  907      0999  6                              SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSCSW_LENGTH];
  908      1000  6
  909      1001  6                              IF .STATE EQL K_SD_T THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
  910      1002  6
  911      1003  6                              IF .TURN EQL 0
  912      1004  6                              THEN
  913      1005  7                                  BEGIN
  914      1006  7                                  SRC_INFO [S_POINTER] = .SOURCE [DSCSA_POINTER];
  915      1007  6                                  END;
  916      1008  6
  917      1009  5                              END;
  918      1010  5
  919      1011  5                          [K_S_NU, K_SD_NU] :
  920      1012  6                              BEGIN
  921      1013  6                              .LEFT_OR_RIGHT_CVT = K_DEC;
  922      1014  6
  923      1015  6                              IF .STATE EQL K_SD_NU THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
  924      1016  6
  925      1017  6                              IF .TURN EQL 0
  926      1018  6                              THEN
  927      1019  7                                  BEGIN
  928      1020  7                                  SRC_INFO [S_LEN] = 31;
  929      1021  7                                  CVTTP (SOURCE [DSCSW_LENGTH], .SOURCE [DSCSA_POINTER], LIBSAB_CVTTP_U,
  930      1022  7                                      SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
  931      1023  6                                  END;
  932      1024  6
  933      1025  5                              END;
  934      1026  5
  935      1027  5                          [K_S_NL, K_SD_NL] :
  936      1028  6                              BEGIN
  937      1029  6                              .LEFT_OR_RIGHT_CVT = K_DEC;
  938      1030  6
  939      1031  6                              IF .STATE EQL K_SD_NL THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
  940      1032  6
  941      1033  6                              IF .TURN EQL 0
  942      1034  6                              THEN
  943      1035  7                                  BEGIN
  944      1036  7                                  SRC_INFO [S_LEN] = 31;
  945      1037  7                                  CVTSP (%REF (
  946      1038  7
  947      1039  7                                      IF .SOURCE [DSCSW_LENGTH] EQL 0 THEN 0 ELSE .SOURCE [DSCSW_LENGTH] - 1),
  948      1040  7                                      .SOURCE [DSCSA_POINTER], SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
  949      1041  7
  950      1042  6                                  END;
  951      1043  6
  952      1044  5                              END;
  953      1045  5
  954      1046  5                          [K_S_NLO, K_SD_NLO] :
```

```
  955    1047  6                        BEGIN
  956    1048  6                        .LEFT_OR_RIGHT_CVT = K_DEC;
  957    1049  6
  958    1050  6                        IF .STATE EGL K_SD_NLO THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
  959    1051  6
  960    1052  6                        IF .TURN EQL 0
  961    1053  6                        THEN
  962    1054  7                            BEGIN
  963    1055  7
  964    1056  7                            BIND FIRST_BYTE = SOURCE [DSC$A_POINTER] : REF VECTOR [,BYTE];
  965    1057  7
  966    1058  7                            SRC_INFO [S_LEN] = 31;
  967    1059  7                            CHSTRANSLATE (LIBSAB_CVT_O_U, .SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], 0,
  968    1060  7                                .SOURCE [DSC$W_LENGTH], TEMP_BUF);
  969    1061  7                            CVTTP (SOURCE [DSC$W_LENGTH], TEMP_BUF, LIBSAB_CVTTP_U, SRC_INFO [S_LEN],
  970    1062  7                                .SRC_INFO [S_POINTER]);
  971    1063  7
  972    1064  7                            IF (.FIRST_BYTE [0] GEQU %X'4A' AND .FIRST_BYTE [0] LEQU %X'52') OR
  973    1065  7                                .FIRST_BYTE [0] EQLU %X'7D'
  974    1066  7                            THEN
  975    1067  7                                BLOCK [.SRC_INFO [S_POINTER] + .SRC_INFO [S_LEN]/2, 0, 0, 4, 0;, BYTE] =
  976    1068  7                                    .BLOCK [LIBSAB_CVTTP_O + .FIRST_BYTE [0], 0, 0, 4, 0;, BYTE];
  977    1069  7
  978    1070  6                            END;
  979    1071  6
  980    1072  5                        END;
  981    1073  5
  982    1074  5                    [K_S_NR, K_SD_NR] :
  983    1075  6                        BEGIN
  984    1076  6                        .LEFT_OR_RIGHT_CVT = K_DEC;
  985    1077  6
  986    1078  6                        IF .STATE EQL K_SD_NR THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
  987    1079  6
  988    1080  6                        IF .TURN EQL 0
  989    1081  6                        THEN
  990    1082  7                            BEGIN
  991    1083  7
  992    1084  7                            LOCAL
  993    1085  7                                SOU_LEN;
  994    1086  7
  995    1087  7                            SOU_LEN =
  996    1088  8                            BEGIN
  997    1089  8
  998    1090  8                            IF .SOURCE [DSC$W_LENGTH] EQL 0 THEN 0 ELSE .SOURCE [DSC$W_LENGTH] - 1
  999    1091  8
 1000    1092  7                            END;
 1001    1093  7                            TEMP_BUF [0, 0, 8, 0] = .BLOCK [.SOURCE [DSC$A_POINTER] + .SOU_LEN, 0, 0, 8, 0;, BYTE];
 1002    1094  7                            CHSMOVE (.SOU_LEN, .SOURCE [DSC$A_POINTER], TEMP_BUF + 1);
 1003    1095  7                            SRC_INFO [S_LEN] = 31;
 1004    1096  7                            CVTSP (SOU_LEN, TEMP_BUF, SRC_INFO [S_LEN], .SRC_INFO [S_POINTER]);
 1005    1097  6                            END;
 1006    1098  6
 1007    1099  5                        END;
 1008    1100  5
 1009    1101  5                    [K_S_NRO, K_SD_NRO] :
 1010    1102  6                        BEGIN
 1011    1103  6                        .LEFT_OR_RIGHT_CVT = K_DEC;
```

```
1012   1104  6
1013   1105  6                          IF .STATE EQL K_SD_NRO THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
1014   1106  6
1015   1107  6                          IF .TURN EQL 0
1016   1108  6                          THEN
1017   1109  7                              BEGIN
1018   1110  7                              SRC_INFO [S_LEN] = 31;
1019   1111  7                              CVTTP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], LIBSAB_CVTTP_O,
1020   1112  7                                  SRC_INFO [S_LEN]; .SRC_INFO [S_POINTER]);
1021   1113  6                              END;
1022   1114  6
1023   1115  6                          END;
1024   1116  5
1025   1117  5                      [K_S_NZ, K_SD_NZ] :
1026   1118  6                          BEGIN
1027   1119  6                          .LEFT_OR_RIGHT_CVT = K_DEC;
1028   1120  6
1029   1121  6                          IF .STATE EQL K_SD_NZ THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
1030   1122  6
1031   1123  6                          IF .TURN EQL 0
1032   1124  6                          THEN
1033   1125  7                              BEGIN
1034   1126  7                              SRC_INFO [S_LEN] = 31;
1035   1127  7                              CVTTP (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], LIBSAB_CVTTP_Z,
1036   1128  7                                  SRC_INFO [S_LEN]; .SRC_INFO [S_POINTER]);
1037   1129  6                              END;
1038   1130  6
1039   1131  5                          END;
1040   1132  5
1041   1133  5                      [K_S_P, K_SD_P] :
1042   1134  6                          BEGIN
1043   1135  6                          .LEFT_OR_RIGHT_CVT = K_DEC;
1044   1136  6
1045   1137  6                          IF .STATE EQL K_SD_P THEN SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSC$B_SCALE];
1046   1138  6
1047   1139  6                          IF .TURN EQL 0
1048   1140  6                          THEN
1049   1141  7                              BEGIN
1050   1142  7                              CVTPS (SOURCE [DSC$W_LENGTH], .SOURCE [DSC$A_POINTER], %REF (31), TEMP_BUF);
1051   1143  7                              CVTSP (%REF (31), TEMP_BUF, %REF (31), .SRC_INFO [S_POINTER]);
1052   1144  7                              SRC_INFO [S_LEN] = 31;
1053   1145  6                              END;
1054   1146  6
1055   1147  5                          END;
1056   1148  5
1057   1149  5                      [K_D_T] :
1058   1150  6                          BEGIN
1059   1151  6                          .LEFT_OR_RIGHT_CVT = K_NBDS;
1060   1152  6                          SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSC$W_LENGTH];
1061   1153  6
1062   1154  6                          IF .TURN EQL 0
1063   1155  6                          THEN
1064   1156  7                              BEGIN
1065   1157  7                              SRC_INFO [S_POINTER] = .SOURCE [DSC$A_POINTER];
1066   1158  6                              END;
1067   1159  6
1068   1160  5                          END;
```

```
1069   1161  5
1070   1162  5              [K_A_BU, K_A_T, K_NCA_BU, K_NCA_T] :
1071   1163  6                  BEGIN
1072   1164  6                  .LEFT_OR_RIGHT_CVT = K_NBDS;
1073   1165  6
1074   1166  7                  IF (.SRC_OR_DST [DSCSL_ARSIZE] GTR K_LRGST_WU OR .SRC_OR_DST [DSCSB_DIMCT] NEQ 1 OR
1075   1167  7                      .SRC_OR_DST [DSCSW_LENGTH] NEQ 1)
1076   1168  6                  THEN
1077   1169  6                      EXITLOOP K_INVNBDS;
1078   1170  6
1079   1171  7                  IF (.STATE EQL K_NCA_BU OR .STATE EQL K_NCA_T)
1080   1172  6                  THEN
1081   1173  7                      BEGIN
1082   1174  7
1083   1175  7                      IF .SRC_OR_DST [DSCSL_S1] NEQ 1 THEN EXITLOOP K_INVNBDS;
1084   1176  7
1085   1177  7                      END;
1086   1178  6
1087   1179  6                  SRC_OR_DST_INFO [M_SCALE] = .SRC_OR_DST [DSCSB_SCALE];
1088   1180  6                  SRC_OR_DST_INFO [M_LEN] = .SRC_OR_DST [DSCSL_ARSIZE];
1089   1181  6
1090   1182  6                  IF .TURN EQL 0
1091   1183  6                  THEN
1092   1184  7                      BEGIN
1093   1185  7                      SRC_INFO [S_POINTER] = .SOURCE [DSCSA_POINTER];
1094   1186  6                      END;
1095   1187  6
1096   1188  5                  END;
1097   1189  5
1098   1190  5              [K_VS_T, K_VS_VT] :
1099   1191  6                  BEGIN
1100   1192  6                  .LEFT_OR_RIGHT_CVT = K_NBDS;
1101   1193  6
1102   1194  6                  IF .TURN EQL 0
1103   1195  6                  THEN
1104   1196  7                      BEGIN
1105   1197  7                      SRC_INFO [S_POINTER] = .SOURCE [DSCSA_POINTER] + 2;
1106   1198  7                      SRC_INFO [S_LEN] = .BLOCK [.SOURCE [DSCSA_POINTER], 0, 0, 16, 0;, BYTE];
1107   1199  7                      END
1108   1200  6                  ELSE
1109   1201  6                      DST_INFO [D_LEN] = .DESTINATION [DSCSW_LENGTH];
1110   1202  6
1111   1203  5                  END;
1112   1204  5
1113   1205  5              [INRANGE] :
1114   1206  5                  LIBSSTOP (LIBS_FATERRLIB);
1115   1207  5              TES;
1116   1208  5
1117   1209  5          END
1118   1210  4      )                                          ! End of INCRU, with a EXITLOOP value.
1119   1211  2      END;                                        ! End of STATUS.
1120   1212  2  !+
1121   1213  2  ! Map the left and right of the conversion, (i.e. if the conversion is
1122   1214  2  ! K_SMLINT_LRGFLT, then LEFT_CVT is SMLINT and RIGHT_CVT is LRGFLT)
1123   1215  2  ! into a final conversion index and return with the status of this routine.
1124   1216  2  !-
1125   1217  2      .CVT_PATH = (.LEFT_CVT - 1)*K_TOT_CAT + .RIGHT_CVT;
```

```
; 1126     1218 2     RETURN .STATUS;
; 1127     1219 1     END;                                             ! End of routine LIBSSFIND_CVT_PATH

;                                                                      .TITLE  LIBSSFIND_CVT_PATH LIBSSFIND_CVT_PATH  for inte
;                                                                                          rnal use of LIBSCVT
                                                                       .IDENT  \1-006\

                                                                       .PSECT  _LIBSCODE,NOWRT,  SHR,  PIC,2

FF  FF  FF  0B  0A  09  FB  FF  FB  FF  04  FF  02  01  FF  00000 P.AAA: .BYTE   -1, 1, 2, -1, 4, -1, -5, -1, -5, 9, 10, -
                                                                               11, -1, -1, -1
OE  FE  FE  0B  0A  09  08  07  06  FE  04  03  02  FE  FE  0000F P.AAB: .BYTE   -2, -2, 2, 3, 4, -2, 6, 7, 8, 9, 10, 11, -
FE  1C  1B  FE  FE  FC  FE  FE  15  14  13  12  11  10  0F  0001E        -2, -2, 14, 15, 16, 17, 18, 19, 20, 21, -
FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FE  FE  FE  FE  FE  0002D        -2, -2, -4, -2, -2, 27, 28, -2, -2, -2, -
FC  FC  FC  FC  FC  FC  FC  0E  FC  FC  FC  FC  FC  FC  FC  0003C        -2, -2, -2, -4, -4, -4, -4, -4, -4, -4, -
FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  0004B        -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -
FE  FE  FD  FD  FD  FD  FD  FD  FE  04  03  02  FE  FE  FE  0005A        14, -4, -4, -4, -4, -4, -4, -4, -4, -4, -
FD  FD  FE  FE  FE  FE  FC  FC  FC  FC  FC  FC  FC  0E  FC  00069        -4, -4, -4, -4, -4, -4, -4, -4, -4, -4, -
FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FE  FC  FE  00078        -4, -4, -4, -4, -2, -2, 2, -3, -3, -2, -
14  13  12  11  10  0F  0E  FC  FC  0B  0A  09  08  07  06  00087        -3, -3, -3, -3, -3, -3, -2, -2, 14, -4, -
FC  FC  FC  FC  FC  FC  FC  1C  1B  FE  FC  FC  FC  FC  15  00096        -4, -4, -4, -4, -4, -4, -4, -4, -2, -2, -
FE  FD  FD  FD  FD  FD  FD  FE  FD  FD  02  FE  FE  FC  FC  000A5        -2, -3, -3, -2, -4, -2, -4, -2, -3, -3, -
FD  FE  FE  FE  FC  FC  FC  FC  FC  FC  FC  FC  0E  FE  000B4        -4, -4, -4, -4, -4, -4, -2, -2, 6, 7, 8, -
FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FE  FC  FE  FD  000C3        9, 10, 11, -4, -4, 14, 15, 16, 17, 18, -
FC  FC  FC  FC  FC  0E  FC  FC  FC  FC  FC  FC  FC  000D2        19, 20, 21, -4, -4, -4, -4, -2, 27, 28, -
FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  FC  000E1        -4, -4, -4, -4, -4, -4, -4, -4, -4, -2, -
                                        25  FC  FC  000F0        -2, 2, -3, -3, -2, -3, -3, -3, -3, -3, -
                                                                               -3, -2, -2, 14, -4, -4, -4, -4, -4, -
                                                                               -4, -4, -4, -2, -2, -2, -3, -3, -2, -4, -
                                                                               -2, -4, -4, -4, -4, -4, -4, -4, -4, -4, -
                                                                               -4, -4, -4, -4, -4, -4, -4, -4, -4, -
                                                                               -4, 14, -4, -4, -4, -4, -4, -4, -4, -
                                                                               -4, -4, -4, -4, -4, -4, -4, -4, -4, -
                                                                               -4, -4, -4, -4, 37

                                    CLASS_TABLE=     P.AAA
                                    DTYPE_TABLE=     P.AAB
                                                  .EXTRN  LIBSSTOP, LIBSSCVT_CVTGH_R1
                                                  .EXTRN  LIBSAB_CVTTP_U, LIBSAB_CVT_O_U
                                                  .EXTRN  LIBSAB_CVTTP_O, LIBSAB_CVT_U_O
                                                  .EXTRN  LIBSAB_CVTPT_U, LIBSAB_CVTPT_O
                                                  .EXTRN  LIBSAB_CVTPT_Z, LIBSAB_CVTTP_Z
                                                  .EXTRN  LIBS_FATERRLIB

                          0FFC 00000               .ENTRY  LIBSSFIND_CVT_PATH, Save R2,R3,R4,R5,R6,R7,-: 0194
                                                          R8,R9,R10,R11
             5E          38 C2 00002               SUBL2   #56, SP
                      0C AE D4 00005               CLRL    TURN                                          0748
   02        00      0C AE CF 00008  1$:           CASEL   TURN, #0, #2                                  0756
   003C      0021        0006   0000D  2$:         .WORD   3$-2$,-
                                                          4$-2$,-
                                                          5$-2$
             50          04 AC D0 00013  3$:        MOVL    SOURCE, R0                                     0761
          04 AE      03 A0 9A 00017               MOVZBL  3(R0), CLASS
          6E         02 A0 9A 0001C               MOVZBL  2(R0), DTYPE                                    0762
          58            50 D0 00020               MOVL    R0, SRC_OR_DST                                  0763
```

```
                              5B      0C  AC  D0 00023           MOVL    SRC_INFO, SRC_OR_DST_INFO           0764
                         08   AE      34  AE  9E 00027           MOVAB   LEFT_CVT, LEFT_OR_RIGHT_CVT        0765
                                      20  11 0002C               BRB     6$                                 0756
                              50      08  AC  D0 0002E 4$:        MOVL    DESTINATION, R0                    0770
                         04   AE      03  A0  9A 00032           MOVZBL  3(R0), CLASS                       0771
                         6E   02      A0  9A 00057               MOVZBL  2(R0), DTYPE
                              58      50  D0 0003B               MOVL    R0, SRC_OR_DST                     0772
                              5B      10  AC  D0 0003E           MOVL    DST_INFO, SRC_OR_DST_INFO          0773
                         08   AE      30  AE  9E 00042           MOVAB   RIGHT_CVT, LEFT_OR_RIGHT_CVT       0774
                                      05  11 00047               BRB     6$                                 0756
                              50      01  D0 00049 5$:           MOVL    #1, STATUS                         0778
                                      5F  11 0004C               BRB     12$
                         0E   04      AE  D1 0004E 6$:           CMPL    CLASS, #14                         0785
                                      05  1B 00052               BLEQU   7$
                              50      05  CE 00054               MNEGL   #5, STATUS
                                      54  11 00057               BRB     12$
                              25      6E  D1 00059 7$:           CMPL    DTYPE, #37                         0787
                                      05  1B 0005C               BLEQU   8$
                              50      06  CE 0005E               MNEGL   #6, STATUS
                                      4A  11 00061               BRB     12$
                              50      04  AE  D0 00063 8$:        MOVL    CLASS, R0                          0792
                              56    FEA1 CF40  98 00067           CVTBL   CLASS_TABLE[R0], STATE
                    12 FFFFFFF9 8F      56  CF 0006D             CASEL   STATE, #-7, #18                    0797
        0035           0035        0035       0035    00075 9$:  .WORD   11$-9$,-
        0026           0035        0035       0035    0007D               11$-9$,-
        003A           003A        003A       003A    00085               11$-9$,-
        003A           003A        003A       003A    0008D               11$-9$,-
                       003A        003A       003A    00095               11$-9$,-
                                                                          11$-9$,-
                                                                          11$-9$,-
                                                                          10$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-
                                                                          13$-9$,-

              00000000G 8F      DD 0009B 10$:        PUSHL   #LIB$_FATERRLIB                 0818
        00000000G 00               01  FB 000A1               CALLS   #1, LIB$STOP
                                   65  11 000A8               BRB     25$
                       50          56  D0 000AA 11$:          MOVL    STATE, STATUS          0801
                                   55  11 000AD 12$:          BRB     23$
                       0A   01     56  CF 000AF 13$:          CASEL   STATE, #1, #10         0806
        002C           0027      0023       000B3 14$:        .WORD   16$-14$,-
        0016           0016      0016       000BB                     17$-14$,-
                       003B      0036       0031    000C3               15$-14$,-
                                                                       18$-14$,-
                                                                       15$-14$,-
                                                                       15$-14$,-
                                                                       15$-14$,-
                                                                       19$-14$,-
```

```
                                                                        20$-14$,-
                                                                        21$-14$
                        00000000G   8F  DD  000C9  15$:   PUSHL    #LIB$_FATERRLIB
                        00000000G   00  01  FB  000CF       CALLS    #1, LIB$STOP
                                    50  D4  000D6  16$:    CLRL     R0
                                    17  11  000D8          BRB      22$
                        50          01  D0  000DA  17$:    MOVL     #1, R0
                                    12  11  000DD          BRB      22$
                        50          02  D0  000DF  18$:    MOVL     #2, R0
                                    0D  11  000E2          BRB      22$
                        50          03  D0  000E4  19$:    MOVL     #3, R0
                                    08  11  000E7          BRB      22$
                        50          04  D0  000E9  20$:    MOVL     #4, R0
                                    03  11  000EC          BRB      22$
                        50          05  D0  000EE  21$:    MOVL     #5, R0
                        50          26  C4  000F1  22$:    MULL2    #38, R0
                        50     FE23 CF40  9E  000F4         MOVAB    DTYPE_TABLE[R0], R0
                        57     00 BE40  98  000FA           CVTBL    @DTYPE[R0], TOKEN
                                    06  18  000FF          BGEQ     24$
                        50          57  D0  00101         MOVL     TOKEN, STATUS
                                073C 31  00104  23$:       BRW      105$
                 50   56           26  C5  00107  24$:    MULL3    #38, STATE, R0
                 56   50           57  C1  0010B          ADDL3    TOKEN, R0, STATE
        0000019F 8F   28           56  CF  0010F  25$:    CASEL    STATE, #40, #415
    0710      0370    0358       0340  00117  26$:   .WORD    27$-26$,-
    03CF      03C0    039B       0376  0011F                   28$-26$,-
    0710      0710    0444       041F  00127                   29$-26$,-
    0545      050C    04DB       04C3  0012F                   102$-26$,-
    0658      0627    05F6       05B1  00137                   30$-26$,-
    0710      0710    0710       0710  0013F                   32$-26$,-
    0710      049D    046F       0710  00147                   35$-26$,-
    0710      0710    0710       0710  0014F                   37$-26$,-
    0710      0710    0710       0710  00157                   43$-26$,-
    0710      0710    0710       0710  0015F                   46$-26$,-
    0710      0710    0710       0710  00167                   102$-26$,-
    0710      0710    0710       0710  0016F                   102$-26$,-
    0710      068A    0710       0710  00177                   57$-26$,-
    0710      0710    0710       0710  0017F                   59$-26$,-
    0710      0710    0710       0710  00187                   63$-26$,-
    0710      0710    0710       0710  0018F                   69$-26$,-
    0710      0710    0710       0710  00197                   76$-26$,-
    0710      0710    0710       0710  0019F                   81$-26$,-
    0710      0710    0710       0710  001A7                   85$-26$,-
    0710      0710    0710       0710  001AF                   89$-26$,-
    0710      0710    0710       0710  001B7                   102$-26$,-
    0710      0710    0710       0710  001BF                   102$-26$,-
    0710      0710    0710       0710  001C7                   102$-26$,-
    0710      0710    0710       0710  001CF                   102$-26$,-
    0710      0710    0710       0710  001D7                   102$-26$,-
    0710      0710    0710       0710  001DF                   50$-26$,-
    0710      0710    0710       0710  001E7                   53$-26$,-
    0710      0710    0710       0710  001EF                   102$-26$,-
    0710      0694    0710       0710  001F7                   102$-26$,-
    0710      0710    0710       0710  001FF                   102$-26$,-
    0710      0710    0710       0710  00207                   102$-26$,-
    0710      0694    0710       0710  0020F                   102$-26$,-
    0710      0710    0710       0710  00217                   102$-26$,-
```

0811

0814

0839

| | | | | | |
|---|---|---|---|---|---|
| 0710 | 0710 | 0710 | 0710 | 0021F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00227 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0022F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00237 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0023F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00247 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0024F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00257 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0025F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00267 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0026F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00277 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0027F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00287 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0028F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00297 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0029F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002A7 | 92$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002AF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002B7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002BF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002C7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002CF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002D7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002DF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002E7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002EF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002F7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 002FF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00307 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0030F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00317 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0031F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00327 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0032F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00337 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0033F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00347 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0034F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00357 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0035F | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 00367 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0036F | 102$-26$,- |
| 0710 | 0370 | 0358 | 0340 | 00377 | 102$-26$,- |
| 03CF | 03C0 | 039B | 0376 | 0037F | 102$-26$,- |
| 0710 | 0710 | 0444 | 041F | 00387 | 102$-26$,- |
| 0545 | 050C | 04DB | 04C3 | 0038F | 102$-26$,- |
| 0658 | 0627 | 05F6 | 05B1 | 00397 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 0039F | 102$-26$,- |
| 0710 | 049D | 046F | 0710 | 003A7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 003AF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 003B7 | 102$-26$,- |
| 0710 | 0694 | 0710 | 0710 | 003BF | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 003C7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 003CF | 102$-26$,- |
| 0710 | 0694 | 0710 | 0710 | 003D7 | 102$-26$,- |
| 0710 | 0710 | 0710 | 0710 | 003DF | 102$-26$,- |

E 1

LIB$FIND_CVT_P LIB$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19     VAX-11 Bliss-32 V4.0-742          Page 26
1-006              Deterministic Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:38:50     [LIBRTL.SRC]LIBFINCVT.B32;1        (3)

```
    0710        0710        0710        0710     003E7              102$-26$,-
    0710        0710        0710        0710     003EF              102$-26$,-
    0710        0710        0710        0710     003F7              102$-26$,-
    0710        0710        0710        0710     003FF              102$-26$,-
    0710        0710        0710        0710     00407              102$-26$,-
    0710        0?.0        0710        0710     0040F              102$-26$,-
    0710        0710        0710        0710     00417              102$-26$,-
    0710        0710        0710        06E7     0041F              102$-26$,-
    0710        0710        0710        0710     00427              102$-26$,-
    0710        0710        0710        0710     0042F              102$-26$,-
    0710        0710        0710        0710     00437              102$-26$,-
    0710        0710        0710        0710     0043F              102$-26$,-
    0710        0710        0710        0710     00447              102$-26$,-
    06E7        0710        0710        0710     0044F              102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                    93$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                    93$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
                                                                   102$-26$,-
```

F 1

LIBSSFIND_CVT_P LIBSSFIND_CVT_PATH   for internal use of LIBSCVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742                    Page 27
1-006            Deterministic Finite Automata for LIBSCVT_DX_DX 14-Sep-1984 12:38:50    [LIBRTL.SRC]LIBFINCVT.B32;1                     (3)

```
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
102$-26$.-
```

G 1

LIBSSFIND_CVT_P LIBSSFIND_CVT_PATH  for internal use of LIBSCVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742                    Page 28
1-006              Deterministic Finite Automata for LIBSCVT_DX_DX 14-Sep-1984 12:38:50    [LIBRTL.SRC]LIBFINCVT.B32;1                      (3)

```
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
```

```
102$-26$,-
1 $-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
27$-26$,-
28$-26$,-
29$-26$,-
102$-26$,-
30$-26$,-
32$-26$,-
35$-26$,-
37$-26$,-
43$-26$,-
46$-26$,-
102$-26$,-
102$-26$,-
57$-26$,-
59$-26$,-
```

1 1

LIB$$FIND_CVT_P LIB$$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19    VAX-11 BLiss-32 V4.0-742           Page 30
1-006              Deterministic Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:38:50    [LIBRTL.SRC]LIBFINCVT.B32;1                (3)

```
63$-26$,-
69$-26$,-
76$-26$,-
81$-26$,-
85$-26$,-
89$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
50$-26$,-
53$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
93$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
93$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
102$-26$,-
```

J 1

LIB$$FIND_CVT_P LIB$$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742           Page 31
1-006                Deterministic Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:58:50       [LIBRTL.SRC]LIBFINCVT.B32;1                (3)

```
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        99$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        102$-26$,-
                                                        99$-26$

        08  BE           01 D0 00457 27$:   MOVL    #1, @LEFT_OR_RIGHT_CVT   0844
                     0C  AE D5 0045B         TSTL    TURN                    0846
                         75 12 0045E         BNEQ    34$
                 51  0C  AC D0 00460         MOVL    SRC_INFO, R1            0848
                 50  04  AC D0 00464         MOVL    SOURCE, R0
        01  B1   04  B0 9A 00468             MOVZBL  24(R0), @1(R1)
                         66 11 0046D         BRB     34$                     0839
        08  BE           01 D0 0046F 28$:   MOVL    #1, @LEFT_OR_RIGHT_CVT   0855
                     0C  AE D5 00473         TSTL    TURN                    0857
                         5D 12 00476         BNEQ    34$
                 51  0C  AC D0 00478         MOVL    SRC_INFO, R1            0859
                 50  04  AC D0 0047C         MOVL    SOURCE, R0
        01  B1   04  B0 3C 00480             MOVZWL  24(R0), @1(R1)
                         4E 11 00485         BRB     34$                     0839
        08  BE           02 D0 00487 29$:   MOVL    #2, @LEFT_OR_RIGHT_CVT   0866
                         57 11 00488         BRB     36$                     0868
```

```
            08   BE          01 D0 0048D 30$:    MOVL    #1, @LEFT_OR_RIGHT_CVT          0877
      0000015C 8F          56 D1 00491          CMPL    STATE, #348                    0879
                           04 12 00498          BNEQ    31$
            68        08 A8 90 0049A          MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)  0881
                      0C AE D5 0049E 31$:    TSTL    TURN
                         57 12 004A1          BNEQ    39$
            51        0C AC D0 004A3          MOVL    SRC_INFO, R1                     0883
            50        04 AC D0 004A7          MOVL    SOURCE, R0
      01    B1        04 B0 98 004AB          CVTBL   @4(R0), @1(R1)
                         23 11 004B0          BRB     34$                              0839
            08   BE          01 D0 004B2 32$:    MOVL    #1, @LEFT_OR_RIGHT_CVT          0890
      0000015D 8F          56 D1 004B6          CMPL    STATE, #349                    0892
                           04 12 004BD          BNEQ    33$
            68        08 A8 90 004BF          MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                      0C AE D5 004C3 33$:    TSTL    TURN                             0894
                         6C 12 004C6          BNEQ    42$
            51        0C AC D0 004C8          MOVL    SRC_INFO, R1                     0896
            50        04 AC D0 004CC          MOVL    SOURCE, R0
      01    B1        04 B0 32 004D0          CVTWL   @4(R0), @1(R1)
                         5D 11 004D5 34$:    BRB     42$                              0839
            08   BE          01 D0 004D7 35$:    MOVL    #1, @LEFT_OR_RIGHT_CVT          0903
      0000015E 8F          56 D1 004DB          CMPL    STATE, #350                    0905
                           5F 13 004E2          BEQL    44$
                         61 11 004E4 36$:    BRB     45$                              0907
            08   BE          02 D0 004E6 37$:    MOVL    #2, @LEFT_OR_RIGHT_CVT          0916
      0000015F 8F          56 D1 004EA          CMPL    STATE, #351                    0918
                           04 12 004F1          BNEQ    38$
            68        08 A8 90 004F3          MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                      0C AE D5 004F7 38$:    TSTL    TURN                             0920
                         73 12 004FA 39$:    BNEQ    48$
            53        0C AC D0 004FC          MOVL    SRC_INFO, R3                     0923
            50        01 A3 D0 00500          MOVL    1(R3), R0
            51        04 AC D0 00504          MOVL    SOURCE, R1
            52        04 A1 D0 00508          MOVL    4(R1), R2
            60           62 D0 0050C          MOVL    (R2), (R0)
            51        04 A0 9E 0050F          MOVAB   4(R0), R1                        0924
            61        04 A2 D0 00513          MOVL    4(R2), (R1)
                         6B 18 00517          BGEQ    49$                              0926
            60           60 D2 00519          MCOML   (R0), (R0)                       0929
            61           61 D2 0051C          MCOML   (R1), (R1)                       0930
      FFFFFFFF 8F          60 D1 0051F          CMPL    (R0), #-1                       0932
                           06 12 00526          BNEQ    40$
            60           60 D4 00528          CLRL    (R0)                             0935
            61           61 D6 0052A          INCL    (R1)                             0936
                           02 11 0052C          BRB     41$                            0932
            60           60 D6 0052E 40$:    INCL    (R0)                             0939
      07    A3          01 88 00530 41$:    BISB2   #1, 7(R3)                        0941
                         7C 11 00534 42$:    BRB     52$                              0839
            08   BE          03 D0 00536 43$:    MOVL    #3, @LEFT_OR_RIGHT_CVT          0950
      00000160 8F          56 D1 0053A          CMPL    STATE, #352                    0952
                           04 12 00541          BNEQ    45$
            68        08 A8 90 00543 44$:    MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                      0C AE D5 00547 45$:    TSTL    TURN                             0954
                         7C 12 0054A          BNEQ    55$
            51        0C AC D0 0054C          MOVL    SRC_INFO, R1                     0956
            50        04 AC D0 00550          MOVL    SOURCE, R0
      01    B1        04 B0 D0 00554          MOVL    @4(R0), @1(R1)
```

L 1

LIB$$FIND_CVT_P LIB$$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742         Page 33
1-006                Deterministic Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:38:50      [LIBRTL.SRC]LIBFINCVT.B32;1              (3)

```
                              7D  11 00559           BRB      56$                                        0839
                  08  BE     03  D0 0055B 46$:        MOVL     #3, @LEFT_OR_RIGHT_CVT                     0963
              00000161  8F   56  D1 0055F             CMPL     STATE, #353                               0965
                              04  12 00566            BNEQ     47$
                  6B     08  A8  90 00568             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
                        0C  AE  D5 0056C 47$:         TSTL     TURN                                      0967
                              67  12 0056F 48$:       BNEQ     56$
                  50     0C  AC  D0 00571             MOVL     SRC_INFO, R0                              0970
                  51     01  A0  D0 00575             MOVL     1(R0), R1
                  50     04  A0  D0 00579             MOVL     SOURCE, R0
                  50     04  A0  D0 0057D             MOVL     4(R0), R0
                  61         60  7D 00581             MOVQ     (R0), (R1)
                              52  11 00584 49$:       BRB      56$                                        0839
                  08  BE     04  D0 00586 50$:        MOVL     #4, @LEFT_OR_RIGHT_CVT                     0978
              00000171  8F   56  D1 0058A             CMPL     STATE, #369                               0980
                              04  12 00591            BNEQ     51$
                  6B     08  A8  90 00593             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
                        0C  AE  D5 00597 51$:         TSTL     TURN                                      0982
                              6A  12 0059A            BNEQ     61$
                  53     0C  AC  D0 0059C             MOVL     SRC_INFO, R3
                  52     04  AC  D0 005A0             MOVL     SOURCE, R2
                  51     01  A3  D0 005A4             MOVL     1(R3), R1
                  50     04  A2  D0 005A8             MOVL     4(R2), R0
              00000000G  00  16 005AC               JSB      LIB$$CVT_CVTGH_R1
                              6D  11 005B2 52$:       BRB      62$                                        0839
                  08  BE     04  D0 005B4 53$:        MOVL     #4, @LEFT_OR_RIGHT_CVT                     0988
              00000172  8F   56  D1 005B8             CMPL     STATE, #370                               0990
                              04  12 005BF            BNEQ     54$
                  6B     08  A8  90 005C1             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
                        0C  AE  D5 005C5 54$:         TSTL     TURN                                      0992
                              6D  12 005C8 55$:       BNEQ     65$
                  51     04  AC  D0 005CA             MOVL     SOURCE, R1
                  50     0C  AC  D0 005CE             MOVL     SRC_INFO, R0
        01  B0     04  B1   10  28 005D2             MOVC3    #16, @4(R1), @1(R0)
                              47  11 005D8 56$:       BRB      62$                                        0839
                  08  BE     06  D0 005DA 57$:        MOVL     #6, @LEFT_OR_RIGHT_CVT                     0998
                  05  AB     68  B0 005DE             MOVW     (SRC_OR_DST), 5(SRC_OR_DST_INFO)          0999
              00000164  8F   56  D1 005E2             CMPL     STATE, #356                               1001
                              04  12 005E9            BNEQ     58$
                  6B     08  A8  90 005EB             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
                        01F8  31 005EF 58$:           BRW      97$                                       1003
                  08  BE     05  D0 005F2 59$:        MOVL     #5, @LEFT_OR_RIGHT_CVT                     1013
              00000165  8F   56  D1 005F6             CMPL     STATE, #357                               1015
                              04  12 005FD            BNEQ     60$
                  6B     08  A8  90 005FF             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
                        0C  AE  D5 00603 60$:         TSTL     TURN                                      1017
                              68  12 00606 61$:       BNEQ     71$
                  50     0C  AC  D0 00608             MOVL     SRC_INFO, R0                              1020
                  05  A0     1F  B0 0060C             MOVW     #31, 5(R0)
                  51     04  AC  D0 00610             MOVL     SOURCE, R1                                1021
        05  A0 00000000G  00  04  B1  61  26 00614   CVTTP    (R1), @4(R1), LIB$AB_CVTTP_U, 5(R0), @1(R0) 1022
                        01  B0     0061F
                              37  11 00621 62$:       BRB      68$                                        0839
                  08  BE     05  D0 00623 63$:        MOVL     #5, @LEFT_OR_RIGHT_CVT                     1029
              00000166  8F   56  D1 00627             CMPL     STATE, #358                               1031
                              04  12 0062E            BNEQ     64$
                  6B     08  A8  90 00630             MOVB     8(SRC_OR_DST), (SRC_OR_DST_INFO)
```

```
                                        OC  AE  D5 00634 64$:    TSTL    TURN                                                    :  1033
                                        76  12 00637 65$:        BNEQ    73$
                               05   AO  OC  AC  D0 00639         MOVL    SRC_INFO, R0                                            :  1036
                                       1F  B0 0063D             MOVW    #31, 5(R0)
                                   51  04  AC  D0 00641          MOVL    SOURCE, R1                                              :  1039
                                        61  B5 00645             TSTW    (R1)
                                        04  12 00647             BNEQ    66$
                                        52  D4 00649             CLRL    R2
                                        05  11 0064B             BRB     67$
                                    52  61  3C 0064D 66$:        MOVZWL  (R1), R2
                                        52  D7 00650             DECL    R2
        01  B0   05  AO   04  B1  52  09 00652 67$:    CVTSP   R2, @4(R1), 5(R0), @1(R0)                                         :  1040
                                        6A  11 0065A 68$:        BRB     75$                                                     :  0839
                                08  BE   05  D0 0065C 69$:       MOVL    #5, @LEFT_OR_RIGHT_CVT                                   :  1048
                            00000167  8F 56  D1 00660           CMPL    STATE, #359                                              :  1050
                                        04  12 00667             BNEQ    70$
                                    6B  08  A8  90 00669         MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                                        OC  AE  D5 0066D 70$:    TSTL    TURN                                                    :  1052
                                        6A  12 00670 71$:        BNEQ    78$
                                    5A  04  AC  D0 00672         MOVL    SOURCE, R10                                             :  1056
                                    59  0C  AC  D0 00676         MOVL    SRC_INFO, R9                                            :  1058
                               05  A9  1F  B0 0067A             MOVW    #31, 5(R9)
00000000G 00        00   04  BA  04  BC  2E 0067E               MOVTC   @SOURCE, @4(R10), #0, LIB$AB_CVT_O_U, -                  :  1059
                               10  AE  04  BC  00689             @SOURCE, TEMP_BUF
   05  A9 00000000G 00   10  AE  04  BC  26 0068D               CVTTP   @SOURCE, TEMP_BUF, LIB$AB_CVTTP_U, 5(R9), -             :  1062
                                    01  B9  00699               @1(R9)
                                    04  BA  9A 0069B             MOVZBL  @4(R10), R1                                            :  1064
                                4A  8F  51  91 0069F             CMPB    R1, #74
                                        06  1F 006A3             BLSSU   72$
                                52  8F  51  91 006A5             CMPB    R1, #82
                                        06  1B 006A9             BLEQU   74$
                                7D  8F  51  91 006AB 72$:        CMPB    R1, #125                                                :  1065
                                        70  12 006AF 73$:        BNEQ    83$
                                50  05  A9  3C 006B1 74$:        MOVZWL  5(R9), R0                                               :  1067
                                    50  02  C6 006B5             DIVL2   #2, R0
                            00000000G0041  9F 006B8             PUSHAB  LIB$AB_CVTTP_O[R1]                                       :  1068
        01  B940        04       00  9E  F0 006BF               INSV    @(SP)+, #0, #4, @1(R9)[R0]
                                        74  11 006C6 75$:        BRB     84$                                                     :  0839
                                08  BE   05  D0 006C8 76$:       MOVL    #5, @LEFT_OR_RIGHT_CVT                                   :  1076
                            00000168  8F 56  D1 006CC           CMPL    STATE, #360                                              :  1078
                                        04  12 006D3             BNEQ    77$
                                    6B  08  A8  90 006D5         MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                                        OC  AE  D5 006D9 77$:    TSTL    TURN                                                    :  1080
                                        74  12 006DC 78$:        BNEQ    87$
                                50  04  AC  D0 006DE             MOVL    SOURCE, R0                                              :  1090
                                        60  B5 006E2             TSTW    (R0)
                                        04  12 006E4             BNEQ    79$
                                    5A  D4 006E6                 CLRL    SOU_LEN
                                        05  11 006E8             BRB     80$
                                    5A  60  3C 006EA 79$:        MOVZWL  (R0), SOU_LEN
                                    5A  D7 006ED                 DECL    SOU_LEN
                            10  AE  04 B04A 90 006EF 80$:        MOVB    @4(R0)[SOU_LEN], TEMP_BUF                               :  1093
                        11  AE  04  B0  5A  28 006F5             MOVC3   SOU_LEN, @4(R0), TEMP_BUF+1                             :  1094
                                50  0C  AC  D0 006FB             MOVL    SRC_INFO, R0                                            :  1095
                                   05  AO  1F  B0 006FF          MOVW    #31, 5(R0)
        01  B0   05  AO   10  AE  5A  09 00703               CVTSP   SOU_LEN, TEMP_BUF, 5(R0), @1(R0)                            :  1096
                                        60  11 0070B             BRB     88$                                                     :  0839
```

N 1

LIB$$FIND_CVT_P LIB$$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742    Page 35
1-006            Deterministic Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:38:50    [LIBRTL.SRC]LIBFINCVT.B32;1    (3)

```
                    08   BE        05 D0 0070D 81$:    MOVL    #5, @LEFT_OR_RIGHT_CVT              1103
               00000169   8F       56 D1 00711         CMPL    STATE, #361                        1105
                                   04 12 00718         BNEQ    82$
                    6B        08   A8 90 0071A         MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                              OC   AE D5 0071E 82$:    TSTL    TURN                               1107
                                   7C 12 00721 83$:    BNEQ    91$
                    50        OC   AC D0 00723         MOVL    SRC_INFO, R0                       1110
                    05        A0   1F B0 00727         MOVW    #31, 5(R0)
                    51        04   AC D0 0072B         MOVL    SOURCE, R1                         1111
05  A0 00000000G 00  04   B1   61 26 0072F         CVTTP   (R1), @4(R1), LIB$AB_CVTTP_0, 5(R0), @1(R0)  1112
                    01   B0             0073A
                    61   11 0073C 84$:    BRB    91$                                0839
                    08   BE        05 D0 0073E 85$:    MOVL    #5, @LEFT_OR_RIGHT_CVT             1119
               0000016A   8F       56 D1 00742         CMPL    STATE, #362                        1121
                                   04 12 00749         BNEQ    86$
                    6B        08   A8 90 0074B         MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                              OC   AE D5 0074F 86$:    TSTL    TURN                               1123
                                   4B 12 00752 87$:    BNEQ    91$
                    50        OC   AC D0 00754         MOVL    SRC_INFO, R0                       1126
                    05        A0   1F B0 00758         MOVW    #31, 5(R0)
                    51        04   AC D0 0075C         MOVL    SOURCE, R1                         1127
05  A0 00000000G 00  04   B1   61 26 00760         CVTTP   (R1), @4(R1), LIB$AB_CVTTP_Z, 5(R0), @1(R0)  1128
                    01   B0             0076B
                    30   11 0076D 88$:    BRB    91$                                0839
                    08   BE        05 D0 0076F 89$:    MOVL    #5, @LEFT_OR_RIGHT_CVT             1135
               0000016B   8F       56 D1 00773         CMPL    STATE, #363                        1137
                                   04 12 0077A         BNEQ    90$
                    6B        08   A8 90 0077C         MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)
                              OC   AE D5 00780 90$:    TSTL    TURN                               1139
                                   77 12 00783         BNEQ    98$
10  AE          1F        04   50   AC D0 00785         MOVL    SOURCE, R0                         1142
                          04   B0   60 08 00789         CVTPS   (R0), @4(R0), #31, TEMP_BUF
                    54        OC   AC D0 00790         MOVL    SRC_INFO, R4                       1143
01  B4          1F        10   AE   1F 09 00794         CVTSP   #31, TEMP_BUF, #31, @1(R4)
                    05        A4   1F B0 0079B         MOVW    #31, 5(R4)                         1144
                                   79 11 0079F 91$:    BRB    100$                               0839
                    08   BE        06 D0 007A1 92$:    MOVL    #6, @LEFT_OR_RIGHT_CVT             1151
                    05   AB        68 B0 007A5         MOVW    (SRC_OR_DST), 5(SRC_OR_DST_INFO)  1152
                                   3F 11 007A9         BRB    97$                                1154
                    08   BE        06 D0 007AB 93$:    MOVL    #6, @LEFT_OR_RIGHT_CVT             1164
               0000FFFF   8F   OC   A8 D1 007AF         CMPL    12(SRC_OR_DST), #65535            1166
                    01        0B   23 14 007B7         BGTR    95$
                              0B   A8 91 007B9         CMPB    11(SRC_OR_DST), #1
                                   1D 12 007BD         BNEQ    95$
                    01             68 B1 007BF         CMPW    (SRC_OR_DST), #1                  1167
                                   18 12 007C2         BNEQ    95$
               0000017E   8F       56 D1 007C4         CMPL    STATE, #382                        1171
                                   09 13 007CB         BEQL    94$
               0000018A   8F       56 D1 007CD         CMPL    STATE, #394
                                   0B 12 007D4         BNEQ    96$
                    01        14   A8 D1 007D6 94$:    CMPL    20(SRC_OR_DST), #1               1175
                                   05 13 007DA         BEQL    96$
                    50             07 CE 007DC 95$:    MNEGL   #7, STATUS
                                   62 11 007DF         BRB    105$
                    6B        08   A8 90 007E1 96$:    MOVB    8(SRC_OR_DST), (SRC_OR_DST_INFO)  1179
                    05   AB   OC   A8 B0 007E5         MOVW    12(SRC_OR_DST), 5(SRC_OR_DST_INFO)  1180
                              OC   AE D5 007EA 97$:    TSTL    TURN                               1182
```

B 2

LIB$$FIND_CVT_P LIB$$FIND_CVT_PATH  for internal use of LIB$CVT 16-Sep-1984 00:54:19    VAX-11 Bliss-32 V4.0-742        Page 36
1-006                Deterministic-Finite Automata for LIB$CVT_DX_DX 14-Sep-1984 12:38:50    [LIBRTL.SRC]LIBFINCVT.B32;1           (3)

```
                               51    0C    45 12 007ED          BNEQ     103$                              ;  1185
                               50    0C    AC D0 007EF          MOVL     SRC_INFO, R1
                         01    A1    04    50 04 AC D0 007F3    MOVL     SOURCE, R0
                                           A0 D0 007F7          MOVL     4(R0), 1(R1)
                                     08    36 11 007FC 98$:     BRB      103$                              ;  0839
                               08    BE    06 D0 007FE 99$:     MOVL     #6, @LEFT_OR_RIGHT_CVT            ;  1192
                                     0C    AE D5 00802          TSTL     TURN                              ;  1194
                                           15 12 00805          BNEQ     101$
                               50    0C    AC D0 00807          MOVL     SRC_INFO, R0                      ;  1197
                               51    04    AC D0 0080B          MOVL     SOURCE, R1
                         01    A0    04    A1 02 C1 0080F       ADDL3    #2, 4(R1), 1(R0)
                               05    AO    04 B1 B0 00815       MOVW     @4(R1), 5(R0)                     ;  1198
                                     18    11 0081A 100$:       BRB      103$                              ;  1194
                               50    10    AC D0 0081C 101$:    MOVL     DST_INFO, R0                      ;  1201
                               05    AO    08 BC B0 00820       MOVW     @DESTINATION, 5(R0)
                                     0D    11 00825 102$:       BRB      103$                              ;  0839
                    00000000G    8F DD 00827 102$:              PUSHL    #LIB$_FATERRLIB                   ;  1206
              00000000G    00    01 FB 0082D                    CALLS    #1, LIB$STOP
                               0C    AE D6 00834 103$:          INCL     TURN                              ;  0748
                         03    0C    AE D1 00837                CMPL     TURN, #3
                                     03 1A 0083B                BGTRU    104$
                               F7C8 31 0083D                    BRW      1$
                               50    01 CE 00840 104$:          MNEGL    #1, STATUS
                         51    34    AE 06 C5 00843 105$:        MULL3    #6, LEFT_CVT, R1                 ;  1217
                         51    30    AE CO 00848                ADDL2    RIGHT_CVT, R1
                    14    BC    FA A1 9E 0084C                  MOVAB    -6(R1), @CVT_PATH
                                     04 00851                   RET                                        ;  1219
```

; Routine Size:  2130 bytes,    Routine Base: _LIB$CODE + 00F3

```
; 1128          1220  1
; 1129          1221  1 END                                      ! End of module LIB$$FIND_CVT_PATH.
; 1130          1222  1
; 1131          1223  0 ELUDOM
```

;
;
;                         PSECT SUMMARY
;
;     Name                     Bytes                      Attributes
;
; _LIB$CODE                    2373  NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)


;                       Library Statistics
;
;                                 -------- Symbols --------    Pages       Processing
;     File                        Total   Loaded   Percent    Mapped      Time
;
; _$255$DUA28:[SYSLIB]STARLET.L32;1    9776      35        0       581       00:00.8

```
;                              COMMAND QUALIFIERS

;      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:LIBFINCVT/OBJ=OBJ$:LIBFINCVT MSRC$:LIBFINCVT/UPDATE=(ENH$:LIBFINCVT
;      )

; Size:           2130 code + 243 data bytes
; Run Time:           00:24.0
; Elapsed Time:       01:37.9
; Lines/CPU Min:      3057
; Lexemes/CPU-Min: 25740
; Memory Used:  433 pages
; Compilation Complete
```
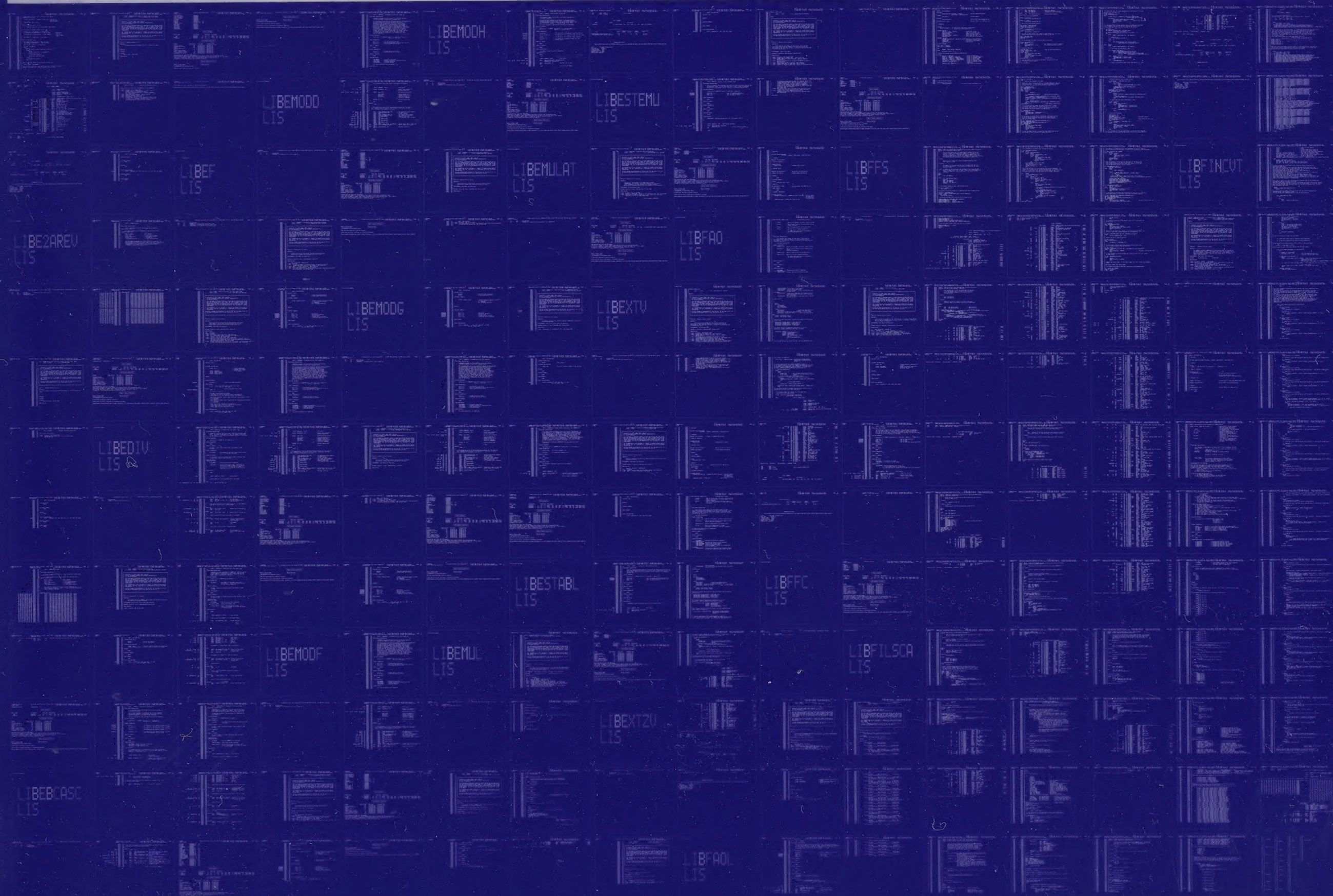
LIBEMODH.LIS

LIBEMODD.LIS

LIBEF.LIS

LIBEMULAT.LIS

LIBESTEMU.LIS

LIBFFS.LIS

LIBFINCVT.LIS

LIBE2AREV.LIS

LIBFAO.LIS

LIBEMODG.LIS

LIBEXTV.LIS

LIBEDIV.LIS

LIBESTABL.LIS

LIBFFC.LIS

LIBEMODF.LIS

LIBEMUL.LIS

LIBFILSCA.LIS

LIBEXTZV.LIS

LIBEBCASC.LIS

LIBFAOL.LIS

LIBFLTUND
LIS

LIBGETSYI
LIS

LIBICHAR
LIS

LIBINITIA
LIS

LIBFIXUPF
LIS

LIBGETFOR
LIS

LIBGETINP
LIS

LIBINISHR
LIS

LIBGETDVI
LIS

LIBGETOPC
LIS

LIBFNDIMG
LIS

LIBGETMSG
LIS

LIBICHAR
LIS

LIBINDEX
LIS

LIBINSQHI

LIBGETJPI
LIS

LIBGETTAB
LIS